Object and human tracking, and robot control through a load sensing floor

Mihai Andries

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Object and human tracking, and robot control through a load sensing floor

THÈSE

présentée et soutenue publiquement le 15 décembre 2015

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Mihai ANDRIES

Composition du jury

Président :	Ye-Qiong Song	Professeur à l'Université de Lorraine
Rapporteurs :	Raja Chatila	Directeur de recherche à l'Université Pierre-et-Marie-Curie
	Anne Spalanzani	Maître de conférences à l'Université Pierre-Mendès-France
Examinateurs :	David Daney	Chargé de recherche à Inria Bordeaux
	Jacques Duchêne	Professeur à l'Université de Technologie de Troyes
	Jan Faigl	Maître de conférences à l'Université Technique de Prague
Directeurs :	François Charpillet	Directeur de recherche à Inria Nancy
	Olivier Simonin	Professeur à l'INSA de Lyon et à l'Université de Lyon

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Acknowledgements

First of all, I thank my thesis advisor, Research Director François Charpillet, as well as my co-advisor, Professor Olivier Simonin, for having offered me the possibility to pursue this doctorate under their guidance. I am also grateful to the head of the "Personally Assisted Living" Inria project lab, Dr. David Daney, who has helped finance this doctoral thesis.

I thank the members of my jury: Pr. Ye-Qiong Song for having presided the jury; Dr. Raja Chatila and Dr. Anne Spalanzani for reviewing my thesis; Dr. David Daney, Pr. Jacques Duchêne and Dr. Jan Faigl for their valuable comments.

I thank our engineers, Dr. Olivier Rochel, Nicolas Beaufort, Thomas Moinel, Dr. Maxime Rio and Mélanie Lelaure, who helped me technically all along this research project. Without them, the projects on which I have worked may not have seen the light. Thank you, Max, for interesting discussions on statistics, as well as for your help with some of the figures in this thesis.

I am grateful to my laboratory colleagues, who have helped me in my search for scientific truth: Dr. Olivier Buffet, for his explanations on Markov Models, Dr. Jilles Dibangoye, for our discussions on scientific theory, Dr. Arseniy Gorin, for his council in mechanics.

I also express my gratitude to the interns that participated in the experimentes related to my thesis: Xavier Erb and Pauline Houlgatte.

My office colleagues, Dr. Benjamin Camus, Dr. Amandine Dubois, Iñaki Fernandez, Abdallah Dib, Nassim Kaldé, Van Quan Nguyen, Julien Vaubourg, Matthieu Zimmer, some of them already doctors, others soon to follow, created a tranquil ambiance, that I shared with them during these years of doctoral studies.

Serena Ivaldi, Francis Colas and Jean-Baptiste Mouret brought fresh blood and dynamism into our new team, LARSEN (Long-term Autonomy and interaction skills for Robots in a Sensing ENvironment), that hosted me after the closure of my previous team, MAIA (Autonomous Intelligent Machines).

I thank our assistants, Véronique Constant, Sophie Drouot and Laurence Félicité, who helped organise all the missions to conferences, project meetings, and summer schools.

My social life during all these years of doctoral research was enriched by interactions with many of my laboratory colleagues. Farewell, Benoît Chappet de Vangel, Mariia Fedotenkova, Meysam Hashemi, Francesco Giovannini, Cecilia Lindig-Leon, Pedro Garcia-Rodriguez, Jirka Maršík, Aleksandre Maskharashvili, Raphaël Cherfan, Valerio Modugno! This list was filled up through the years, but I realise that I may have forgotten a lot of people. Please excuse me for that.

At last, I thank the members of my family for their love and comprehension of my lifestyle during the last phase of my student life, which was this doctorate. Thank you, Zhanna, for your support throughout my years of studies.

Contents

Part II Services provided by sensing floors

Chapter 5
Object surface pressure scanning

Chapter 6
Detecting, tracking and identifying humans, robots and objects

Part III Integration of sensing floors with personal assistant robots

Chapter 7
Providing roadmaps for autonomous robotic navigation

Overture

1

Introduction

Contents

1.1 Ambient Intelligence

Ambient Intelligence is a branch of *Artificial Intelligence*, which is centered on autonomous perception and actuation inside delimited environments. It is charged with the supervision and control of the entrusted space, monitoring events and human well-being, recognizing human activities through a network of heterogeneous sensors, and providing support through its robotic actuators. *Ambient Intelligence* may be applied in domestic environments — to allow safe ageing at home, in healthcare facilities (hospitals and retirement homes) — to continually diagnose the inhabitants and detect emergencies, in transportation — to control the comfort level inside cabins.

Among the various types of sensors employed in the perception of the environment, load sensors can measure the forces exerted by humans and objects on their surroundings. Floors equipped with such sensors can sense things and living beings by their weight profile. This thesis will explore and analyse the capabilities of such load-sensing floors. More formally, it will explore the paradigms applicable for the analysis of sensing information coming from an omnipresent floor-pressure sensor, located inside a delimited physical space.

As domain of application, we will center on assisted living facilities for elderly people, to allow them to age comfortably at home, and also to reduce the workload of the medical personnel in hospitals, retirement homes and similar facilities. However, these developments are not domain-specific, and can be applied for generic systems managing environments.

1.2 Ageing societies: national problems

Population ageing is a consequence of longer life expectancy, which increases the number of dependent people over the age of 65, and lower birth rate, which decreases the number of economically active people in the next generation. The potential support ratio (the ratio of people aged 25-64, and those over 65) for Western Europe stood at 3.6 in 1995, is currently at 2.7 in 2015, and is projected to reach 1.9 by 2030, and 1.6 by 2050, according to the world population prospects of the United Nations Department of Economic and Social Affairs [186]. Countries like Germany, Austria, France, Belgium, and the U.K. are expected to be among the most affected (see Table 1.1). The population projections for two of the most affected countries, Japan and Germany, are shown in Figures 1.1 and 1.2. If these low potential support ratios will prove to be economically unsustainable, this may impose more stress on the economically active part of the society, which will have to provide for the non-active members.

Table 1.1 – The evolution and projection (medium variant) of the potential support ratio (ratio of population aged 25-64 per population 65+) for a selection of countries. Source: United Nations Department of Economic and Social Affairs [186]

	Potential support ratio			
Country	**1995**	**2015**	**2030**	**2050**
Austria	3.6	2.9	2.1	1.5
China	8.3	6.3	3.3	1.8
France	3.4	2.7	2.0	1.7
Germany	3.7	2.6	1.8	1.4
Japan	3.8	1.9	1.6	1.2
Republic of Korea	8.9	4.5	2.3	1.3
Russian Federation	4.3	4.4	2.7	2.4
Singapore	9.0	5.1	2.3	1.4
Switzerland	3.8	3.1	2.2	1.6
United Kingdom	3.3	2.9	2.3	1.9
United States of America	4.1	3.6	2.4	2.2

1.3 Economical impact of population ageing on societies

The economical impact of population ageing can be estimated by analysing the total expenditure in the sectors adressing the needs of the elderly (e.g. senior housing, assisted living), which are now collectively referred to as the *Silver Economy* [142].

According to a 2013 report [86] made by the National Center for Health Statistics, the annual expenditures for *long-term care services* in the United States of America are estimated to be between $210.9 billion and $306 billion. Long-term care services include assistance with activities of daily living (e.g. dressing, bathing, toileting), instrumental activities of daily living (e.g. medication management and housework), and health maintenance tasks. The mentioned

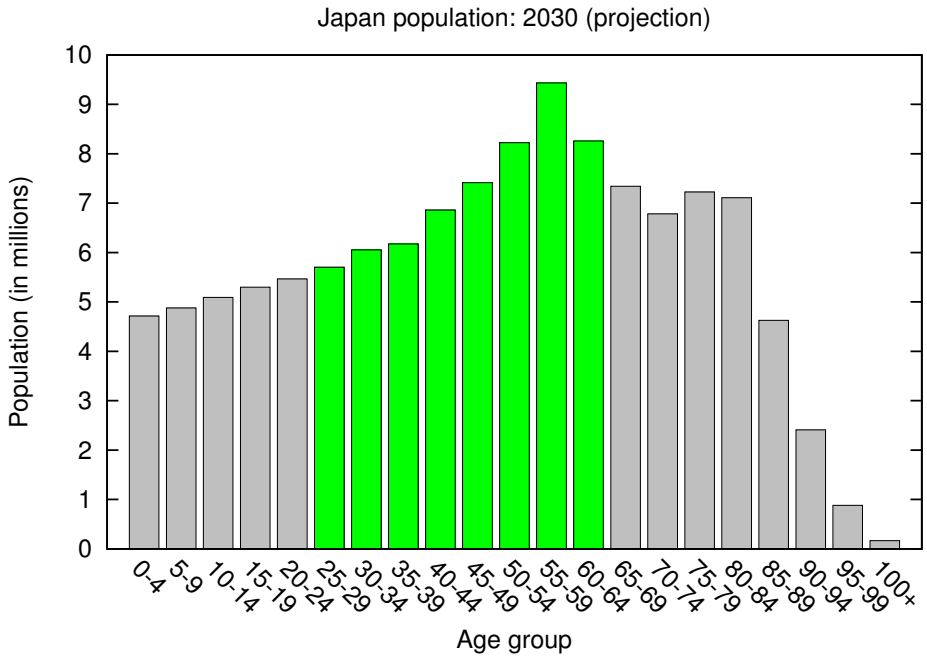


Figure 1.1 – Population projection for Japan, 2030, medium fertility variant. The economically active part of the society is highlighted in green. Source: United Nations Department of Economic and Social Affairs [186]

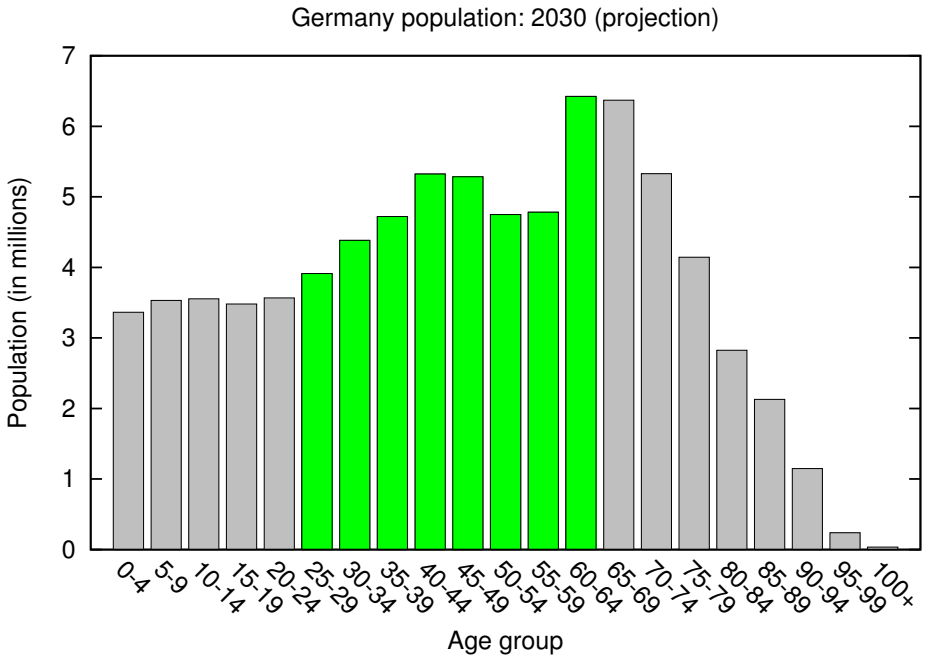


Figure 1.2 – Population projection for Germany, 2030, medium fertility variant. The economically active part of the society is highlighted in green. Source: United Nations Department of Economic and Social Affairs [186]

expenditures include services provided by home health agencies, adult day services centers, assisted living communities, and institutions like nursing homes.

In the European Union, according to a 2008 report by Ernst & Young [43], the public expenses on senior housing equalled €8.7 billion in Italy (2005), €10.3 billion in the United Kingdom (2004), €7.6 billion in Germany, and €1.32 billion in Belgium (2003). In France, the market size of the personal assistance services was evaluated at €12 billion in 2007.

According to a report by KPMG dating from 2013 [77], the average duration of stay in a retirement home in France is 46 months, with an average age of 85 for the internees. The average monthly cost of living inside a retirement home in France was 1857 euros in 2013.

1.4 The search for technical solutions for senile care

A part of the costs related to human assistance can be reduced through the automatisation of the relevant tasks (e.g. continuous diagnosis of health-state, surveillance and fall detection, assistance with fetching things). An environmentally-embedded ambient intelligence could reconstruct a model of the environment that would be used for spatial reasoning, perceive the events taking place inside it through a network of sensors, and react to changes by using its robotic actuators — assistant robots. The assistant robots would also perform the physical interaction tasks for the humans that they serve. This can be conceptually seen as living inside a robot-house.

The search for technical solutions to these problems is an opportunity to advance the state-of-the-art in ambient intelligence and adjacent fields of artificial intelligence and robotics, such as computer vision, multi-modal sensing, object recognition, manipulation of objects, robotic navigation in populated environments, and activity recognition.

Although the automatisation of these personnal assistance tasks would cut costs by reducing the need for constant supervision by medical personnel, it also has its weaknesses. The sensor network necessary for perceiving the environment requires hardware and installation costs, as well as occasional maintenance. Its cost is also directly proportional to the size of the supervised environment, which influences the complexity of the sensor network required to cover it.

1.5 Practical aspects of the introduction of ambient intelligence systems

Ambient intelligence technology could be employed both for the automatisation of retirement homes, as well as for enabling safe ageing at home. In conditions of private use, these technologies for ageing at home could be used by every generation. The perceived value of the technological solution being indicated by its rate of use, economically viable ageing-at-home solutions should be low cost. More complex and expensive technological solutions could become economically viable when used for a longer period of time, as when integrated into a more general home-surveillance system, including applications for child security. Such technical solutions could be either bought, or used on long-term leasing.

On the other hand, healthcare institutions, like retirement homes, which are guaranteed to intensively exploit such systems for senile care, could find it rational to acquire more sentient, robust and complex solutions. Quicker reaction times in cases of emergency would be guaranteed by a constant supervision by the ambient intelligence system. The detection of falls allows to react faster, which is important because the bigger is the time delay between the fall and the medical intervention, the heavier are the consequences for the victim. Not only would automatic

supervision enhance the safety levels in healthcare institutions, it would also reduce the number of human employees performing safety rounds, as well as diagnose patients' health-state.

1.6 Ambient Intelligence projects around the world

Numerous projects are developed all over the world on the topic of remote domestic medical surveillance (also called monitoring). They aim to test a remote surveillance system on a specific category of patients (people with heart or pulmonary failure, asthma, diabetes, Alzheimer's disease, etc.) or to conceive habitats with home automation, or sensors to install in a habitat or worn by a person, or alarm systems adapted to the requirements of remote medical surveillance. Several examples of projects are listed below, sorted by their geographical location.

In France, Thomesse [161] have developed the Tissad project, that has the objective to define a generic architecture for remote monitoring systems, that would be modular and open. Diatélic [162] has developed systems for remote medical monitoring of dialysed persons in their homes, an effort in which the Inria teams Temps réel et interopérabilité (Real time and interoperability) (TRIO) and Machines Intelligentes Autonomes (MAIA) have participated.

The Casper project [111] creates tools for intelligent monitoring of elderly people, and for their communication with their families and healthcare professionals. In particular, Casper aims do develop an innovative system which would be able to follow the domestic activities of elderly or handicapped people, using specific autonomous sensors, so as to provide solutions that would improve their life at home through an adapted behavioural analysis and social assistance. The Prosafe project [31] targets patients suffering from Alzheimer's disease. The Actidom project (Domestic Activity monitoring) [36] has the objective to measure the activity of fragile elderly persons in their day-to-day life, so as to determine the evolution of their state of dependence. The Gerhome [113], CIU-santé (CIU-health) [33, 64], Sweethome [127], and ParaChute projects [60] have also tackled the problem of an automated, objective evaluation of the frailty of elderly persons, using fixed cameras and experimental sensors

In the United States of America, the University of Colorado at Boulder has developed the Adaptive Control of Home Environments (ACHE) project [97] to automatically control the temperature, heating and the lighting levels inside a building. The system supervises the environment, observes the actions taken by the residents on the lights and thermostats, and then constructs a model that allows to predict their future actions, using reinforcement learning based on neural networks. The projects MavHome [38] of the University of Arlington, House of Matilda [57] and Gator Tech Smart House of the University of Florida [71], Aware Home Research [58] of the Georgia Institute of Technology are similar projects having the objective to optimise the comfort and security of elderly people at home.

Microsoft has also proposed a system for following residents, developed for the EasyLiving project [25]. The system uses cameras coupled to movement detectors placed on the walls for tracking and locating humans through image processing techniques. Finally, the Hat project [45] was developped for people suffering from asthma.

In Asia, at the University of Ibaraki in Japan, an intelligent environment was developed, called SELF (Sensorized Environment for LiFe) [101]. It aims to supervise the state of health of persons, by analysing several physiological criteria. The system registers and analyses the physiological data, and then provides a daily report on the physiological activities of the concerned person. The systems developed by the University of Tokyo [102] and the intelligent house of Osaka [89] are other Japanese examples of intelligent homes similar to SELF. In South Korea, the Intelligent Sweet Home project [79] proposes an intelligent house dedicated to persons with reduced mobility.

The goal is to assist persons with going to bed, sitting down in an armchair, and so on.

In Europe, the CarerNet system [184] has been proposed in the United Kingdom, with the aim of offering several healthcare services at home, such as the alarm, tracking of health procedures, and e-health. In the Netherlands, the Senior Citizens Technology Centre has equipped a house with a system for surveilling people, and with assistive technologies [175]. Movement detectors measure the activity of individuals and signal all suspicious inactivity and intrusions.

In Spain, a platform for domestic medical care has been developed, to aid patients suffering from specific pathologies to live inside their homes. The platform was composed of two parts: a client application for the local processing of data, and a medical call-center [56]. Rodriguez *et al.* [128] have developed a generic architecture of a remote medical surveillance system for the European Prototype for Integrated Care project (EPIC).

In Norway, the SmartBo house [42] was built specifically for elderly people. The main elements of the house are controlled automatically: the lighting, doors, windows, window blinds, etc. In Finland, an automated house environment called Terva [76] has been developed to simultaneously supervise several psycho-physiological criteria, exploiting physiological measurements and the behavioural states of persons. The intelligent houses of British Telecom and Anchor Trust [14] in England are other examples of this type of platforms in Europe, having the goal to remotely survey and measure the activity of elderly people.

Efforts are currently made to integrate ambient intelligence solutions into habitats and buildings in their design and construction phases, to avoid the more expensive retro-fitting. Together with the Office d'Hygiène Sociale (OHS) (Office of Social Hygiene) [1] and Pharmagest Diatelic [2], we are developing a project for designing and building a retirement home equipped with an ambient intelligence system, exploiting complementary technologies like non-intrusive load-sensing floors, depth cameras, and mobile assistant robots which act as sensing platforms for inspection of space outside of the system's surveillance area (i.e. in case of occlusions by furniture).

Privacy issues are avoided by automatically processing all the perceived sensing data on the spot, without transmitting sensitive data outside the local network. Outside help is sought in case of emergency, or whenever there is the need for it.

1.7 Background of the thesis

This thesis places itself within the Personally Assisted Living (PAL) initiative, and was financed by an Inria CORDI-S grant. The Large Scale initiative Action PAL is a research infrastructure that proposes technological solutions to improve the quality of life of frail people and their immediate family and caregivers. These solutions come in the form of smart sensors, robotics, and home automation systems. They aim to improve the security and the autonomy of elderly, allow them to remain in their own homes, and to preserve or restore their daily-life necessary functions.

The PAL initiative gathers 9 Inria teams associated with 6 research partners (technological, medical or social), which work together on three main issue guidelines: mobility assistance, assessing the degree of frailty of the persons, and home activities analysis. The systems proposed by the teams communicate with each other, in order to exchange information about the persons and their environment. This constitutes a sensors/actuators network, which is adapted to the physical and cognitive situation of the person.

[1] Office d'Hygiène Sociale: http://www.social.nancy.fr/pagint/fiche.php?cid=464
[2] Pharmagest Diatelic: http://www.diatelic.com/

Our research team has been involved in several actions in the domain of assistance technologies: the PAL project on the development of technologies for assisted living, the Agence Nationale de la Recherche (French National Research Agency) (ANR) Parachute project on fall prevention, the Contrat de plan État-Région (planning agreement between the national and regional governments) (CPER) InfoSitu project on pervasive computing, the Living Assistant Robot (LAR) project on the development of a low-cost navigation system for a robot evolving in an indoor environment, the Whole-Body Compliant Dynamical Contacts for Cognitive Humanoids (CoDyCo) project on the control and cognitive understanding of robust, goal-directed whole-body motion interaction with multiple contacts, and the SATELOR project on providing autonomous medical surveillance at home (a project). This has resulted in several doctoral theses:

- Development of a system for passive 3D tracking of human movement through particle filtering (by Jamal Saboune, 2008),
- Stochastic modeling for medical reasoning and its applications in e-health (by Cédric Rose, 2011),
- Frailty measurement and fall detection for maintaining elderly people at home (by Amandine Dubois, 2014),
- Object and human tracking, and robot control through a load sensing floor (by Mihai Andries, 2015),
- Robust posture tracking using a depth camera (by Abdallah Dib, ongoing work).

1.8 Structure of the thesis

The scientific objective of this thesis is to explore sensing floors' capabilities as a sensor for ambient intelligence: both for creating a representation of the events happening inside the supervised scene, and as an artificial environment for supporting robotic navigation and exploration. This thesis, which constitutes this scientific exploration, is structured in 3 parts, organised as follows.

Part I is dedicated to a general presentation of load-sensing floors. In Chapter 2, we provide an overview of the existing sensing floors, and of the methods developed for human recognition. Chapter 3 presents the sensing floor prototype developed at Inria Nancy, which was used throughout this thesis. Chapter 4 introduces the software developed for processing the data generated by our sensing floor, the sensor calibration procedure, and analyses the possibility of simulating such a sensing floor.

Part II presents our contributions on the use of sensing floors for perception of the environment. Chapter 5 introduces a new technique for scanning the pressure distribution inside the surface of objects in contact with the ground by using sub-pixel shifting. Chapter 6 presents a new method for tracking and localising objects on load-sensing floors, using a mix of inference and combinatorial search techniques.

Part III presents novel ways of integrating sensing floors with ambient intelligence systems, to provide navigation services for assistant robots (Chapter 7), as well as environment-embedded memory capacities for stigmergic exploration algorithms (Chapter 8). A general conclusion is drawn in Chapter 9, together with perspectives for future research.

1.9 Key contributions of this thesis

This thesis provides several contributions, which consist in innovative approaches for using a load-sensing floor. We develop new techniques, some inspired from the field of computer vision, for perceiving objects and activities happening on the floor. We also develop new ways of interaction between robots and the ambient intelligence environment, by using the capabilities of a sensing floor. More precisely, we provide in this thesis:

- a method for high-resolution pressure sensing of objects on a low-resolution load-sensing floor (Chapter 5),
- a technique for localising and recognising objects on a load-sensing floor (Chapter 6),
- an aid to robotic navigation, by generating and providing the safest navigation paths through the environment using sensing floors (Chapter 7),
- an algorithm for distributed robotic exploration of environments with embedded memory capacities (Chapter 8).

1.10 Publication list

This thesis contains parts that have been published in international journals, or presented at international conferences, as well as extracts from yet unpublished, ongoing work.

Journal articles

- Localisation of humans, objects and robots interacting on load-sensing floors.
  Mihai Andries, Olivier Simonin, François Charpillet
  IEEE Sensors Journal, 2015

Conference articles

- High resolution pressure sensing using sub-pixel shifts on low resolution load-sensing tiles.
  Mihai Andries, François Charpillet, Olivier Simonin
  Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA 2015)
- Multi-robot taboo-list exploration of unknown structured environments.
  Mihai Andries, François Charpillet
  Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)
- Multi-robot exploration of unknown environments with identification of exploration completion and post-exploration rendez-vous using ant algorithms.
  Mihai Andries, François Charpillet
  Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)

Papers in preparation

- Roadmap extraction service provided by a sensing floor network for robotic navigation.
  Nassim Kaldé, Mihai Andries, François Charpillet, Olivier Simonin

- Probabilistic sensor data processing for robot localisation on load-sensing floors.
  Maxime Rio, Francis Colas, Mihai Andries, François Charpillet

Part I

Sensing floors

2

Sensing floors: existing prototypes and related work

Contents

2.1 Sensing floors as sensors for ambient intelligence

Sensing floors are floors equipped with sensors for various practical applications, like perception of the environment, health-monitoring, security, and entertainment. Most of these challenges are related to the field of ambient intelligence, in which an artificial entity perceives, reasons, and controls an environment placed under its responsibility.

The use of floor-sensors in ambient intelligence contexts began in the late 1990's, with projects like the ORL active floor [3] by Addlesee *et al.* , the Magic carpet [105] by Paradiso *et al.*, and the Smart floor [104] by Orr and Abowd, where they provided information for reasoning about the observed space. These floors were later on integrated into smart environments, aimed at delivering assistance services like continuous diagnosis of users' health. These smart environments also integrated assistive robotic technologies with sensing networks. Examples of such environments include the Gator Tech Smart House made by the University of Florida [58], the Aware Home introduced by the Georgia Institute of Technology [71, 104], the RoboticRoom system [95, 136] developed by the University of Tokyo, and the ActiSen activity-aware sensor network [39] jointly developed by researchers from several US universities. Further advances were brought by research on human footprint biometrics [100, 121, 78], which allowed sensing floors to recognise humans by extracting features from footprints and human gait.

This chapter first presents the strengths and weaknesses of sensing floors, and then continues on to a listing of the existing floor prototypes, together with the applications developed by various research groups on these floor sensors.

2.2 Advantages of sensing floors

Sensing floors present a series of advantages compared to other types of sensors, both in terms of privacy and perception. Perception happens in the plane of the floor, allowing a direct detection of objects on its surface. They do not suffer from occlusion, as in the case of vision-based cameras. Load-sensing floors can record the pressure profiles of static and dynamic entities (objects and living beings) evolving on the floor. The pressure profiles allow to track, localise and recognise these entities. We can make assumptions about the conservation of weight in a scene (except when entities enter or exit the scene), which helps in tracking the number of objects present in the scene. Compared to vision cameras that suffer from distinct lighting conditions and shadows, sensing floors can simply work in the dark. They are also less intrusive, providing sensing information in a format which is not natural for human understanding.

2.3 Disadvantages of load-sensing floors

At the same time, being installed inside or under the floor, the load sensors also have some downsides. They perceive only a projection of the forces involved in human daily activities, which leaves space for ambiguities in weight sensing, and subsequent tracking and recognition. Thus, whenever floor sensors are insufficient for any of the three tasks (localisation, tracking and identification), additional sensors can be used to solve emerging data ambiguities from a multi-modal perspective. Sensing floors have been combined with radio-frequency identification (RFID) systems [94], pyroelectric infrared sensors (PIR) (for calculating the body surface area) [4], wearable accelerometers [146, 63], audio capture systems [147], and cameras [29, 191]. As the sensors are under regular strain, the precision of their measurements degrades over time.

2.4 Existing prototypes of sensing floors

In this section, we present the developed sensing floor prototypes in chronological order, subdividing them into floors with binary pressure-sensing capacity, and floors with the capacity to measure the intensity of the applied pressure. Each category is futher subdivided into monolithic and modular prototypes. We also present in a separate category several prototypes of sensing floors that perceive the environment through forces other than pressure (section 2.4.3). A compact listing of the floor prototypes described in the related literature is given in Table 2.1. In another table at the end of this section, Table 2.2, we list the floors on which more advanced applications have been developed, such as multi-target tracking and human recognition, together with the features they extract and the methods they employ for human and object recognition. This table updates the lists previously presented in [116] and [171].

2.4.1 Binary pressure-sensing floors

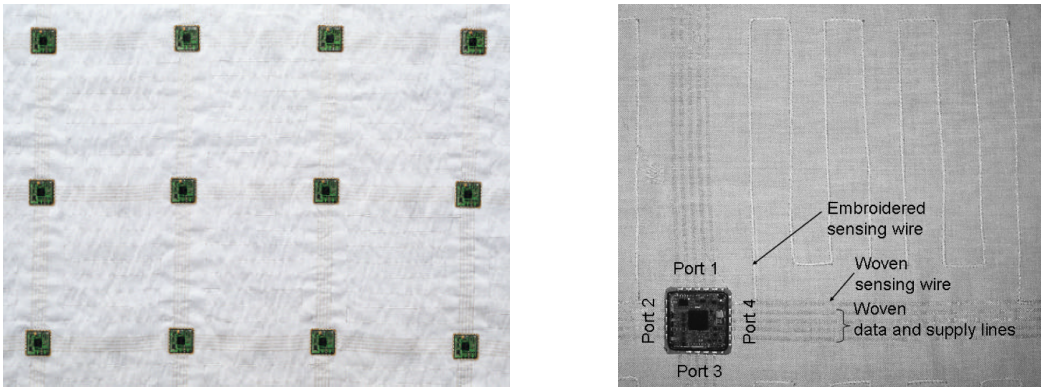
Monolithic binary pressure-sensing floors

Griffith and Fernström developed a sensing floor that contained a matrix of optical proximity sensors, and which was used as interface for musical instruments [55]. The prototype had a

surface of 1.76 m^2, with a pressure-sensing resolution of 40 mm.

Middleton *et al.* [91] designed a 3 m by 0.5 m sensor mat, consisting of perpendicular wires held appart by foam, which act as binary pressure switches (see figure 2.3). They identified people by their stride length, stride cadence, and time-on-toe to time-on-heel ratio, using a standard distance metric of similarity.

Glaser *et al.* developed a textile-based large-area sensor network, integrated into a carpet of size 2.4 m by 2 m [53]. Flexible textile-based conductive material is shaped into 15 cm by 15 cm constructions (see figure 2.1). The carpet could be produced in a reel-to-reel process, and cut in arbitrary shapes. Savio and Ludwig used this floor for detecting footsteps and calculating user trajectories [137].

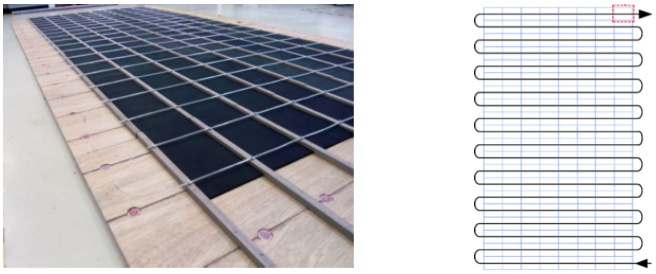


(a) Microprocessor modules integrated into a fabric with interwoven silver-plated copper wires.

(b) An interwoven sensor and a node on the fabric.

Figure 2.1 – The sensing carpet presented by Glaser *et al.* in [53].

Shen and Shin developed a sensing floor using an optical fiber sensor, which employed Brillouin optical correlation domain analysis (BOCDA) to detect the strain deviation along a fiber caused by pressure events [141] (see figure 2.2). A 40 m fiber was embedded between two layers of soft material, forming a sensing surface of size 1.6 m by 4 m. The floor was divided into 160 blocks, each of size 20 cm by 20 cm. The resolution of the floor could be changed by modifying the placement of the optical fiber under the floor. With the suitable data-processing algorithms, the floor was able to detect and track human occupants.



(a) The sensing floor prototype

(b) Structure of the sensing floor

Figure 2.2 – A sensing floor which employs BOCDA technology, presented by Shen and Shin in [141].

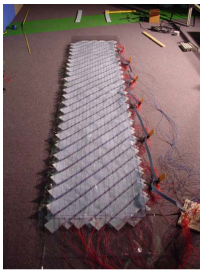


Figure 2.3 – The sensor mat presented by Middleton *et al.* in [91].

Table 2.1 – Load-sensing floors and the sensing technologies employed

Authors	**Publ. Year**	**Floor sensor**	**Modular**
Reilly *et al.* [122]	1991	magnetorestrictive delay lines	✓
Pinkston [108]	1994	FSR	
Addlesee *et al.* [3]	1997	strain gauge load cells	✓
Paradiso *et al.* [105]	1997	piezoelectric wire sensors	
Griffith and Fernström [55]	1998	optical proximity sensors	
Orr and Abowd [104]	2000	strain gauge load cells	✓
Schmidt *et al.* [138]	2002	strain gauge load cells	
Morishita *et al.* [96]	2002	pressure switches	✓
Pirttikangas *et al.* [110, 109, 152, 150, 153]	2003	EMFi	✓
Yun *et al.* [192]	2003	on/off switch sensors	✓
Jung *et al.* [65, 66]	2003	pressure mats	✓
Yin and Pai [190]	2003	pressure mat	✓
Leikas *et al.* [82]	2003	N/A	✓
Lee *et al.* [81]	2004	FSR	✓
Murakita *et al.* [98]	2004	pressure switch sensor	✓
Richardson *et al.* [123]	2004	FSR sensors	✓
Middleton *et al.* [91]	2005	FSR mats	
Yun *et al.* [194, 195, 193]	2005	photo interrupter sensors	✓
Glaser *et al.* [53, 137]	2007	sensing wire	
Suutala *et al.* [154]	2008	pressure switch sensors	✓
Rangarajan *et al.* [119, 120, 115, 116]	2008	FSR mats	✓
Bose and Helal [23]	2008	piezoelectric force sensors	✓
Wen-Hau *et al.* [182]	2008	N/A	✓
Vera-Rodriguez *et al.* [173, 170, 171]	2009	piezoelectric force sensors	✓
Shen and Shin [141]	2009	optical fiber sensor	
Visell *et al.* [177, 179]	2010	FSR	✓
Anlauff *et al.* [11]	2010	FSR	✓
Chang *et al.* [32]	2010	piezoresistive force sensor	✓
Klack *et al.* [73]	2011	piezoelectric force sensors	✓
Lombardi *et al.* [85]	2013	conductive polymer between aluminium stripe electrodes	✓
Al-Naimi *et al.* [4]	2014	FSR	✓
Heller *et al.* [59]	2014	strain gauge load cells	✓
Inria SmartTiles [9]	**2015**[a]	**strain gauge load cells**	✓

[a] Floor built in 2012, article published in 2015.

Modular binary pressure-sensing floors

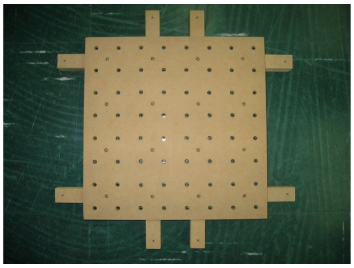
Reilly *et al.* developed an imaging walkway capable of sensing foot-to-ground contact for gait analysis using magnetorestrictive delay lines [122]. This technology was originally developed for memory storage in early computers. The prototype was a modular floor, consisting of panels 0.32 m by 0.64 m, assembled into a 2.5 m by 0.64 m walkway.

Morishita *et al.* presented a modular high-resolution floor sensor, composed of 0.5 m square tiles, each equipped with 4096 pressure switches in a 64x64 array [96], which provided binary information about the presence or absence of load on them. Its high resolution allowed to obtain sharp images of the surfaces in contact with the floor, such as footprints or shoe soles, suggesting that automatic distinction between walking barefoot and in shoes should be possible.

Murakita *et al.* [98] used the InfoFloor system VS-SS-F, developed by Vstone Corporation from Osaka, Japan. The floor has 1140 binary pressure sensors, of size 18 cm by 18 cm. The authors used a Markov Chain Monte Carlo (MCMC) method to perform human tracking. The prediction model was modeled in 2 ways: a generic linear model for predicting the position at time t (using the position at time $t-1$, the current velocity, and a Gaussian noise), and a bipedal model of sensor activations while walking (based on which foot was put forward). Tracking would fail whenever two or more targets crossed their paths, generating tracking ambiguity. Attempts were made to solve this problem by fusing the information from the floor sensors with that from on-body acceleration sensors [63].

Yun *et al.* worked on several sensing floor prototypes. Their first prototype, the UbiFloorI [192] consisted of 144 on/off switch sensors fitted onto a cushioned carpet, with a walking area of 3 m by 1 m (see figure 2.4a). The UbiFloorII [194, 195, 193] was a 12x2 array of wooden tiles, each measuring 30 cm x 30 cm and each containing 64 uniformly-distributed photo-interrupter sensors (see figure 2.4b). These floors were used for user identification based on the extracted gait features (stride length, foot angle, heel strike time, etc.).

(a) UbiFloorI, presented by Yun *et al.* in [192].



(b) The UbiFloorII presented by Yun *et al.* in [194, 195, 193], and a close-up of a tile.

Figure 2.4 – The UbiFloor series of prototypes

Suutala *et al.* used Gaussian Process Joint Particle Filtering to track multiple humans on a tiled floor equipped with binary switch sensors (a VS-SF55 InfoFloor sensor system made by Vstone Corporation, Japan) [151]. The floor contained 12 tiles of size 0.5 m by 0.5 m, and each tile included 25 binary switch sensors of size 10 cm by 10 cm. The floor data was collected at a 16 Hz sampling rate. They also performed person identification on this floor, using gait features and Gaussian Process classification [154].

2.4.2 Floors made of pressure intensity sensors

In the case of floors measuring pressure intensity, we can also exploit the weight information to evaluate the generated tracking hypotheses, and to recognise the entities located on the floor. This section introduces examples of such floors.

Monolithic pressure-sensing floors

Pinkston developed a pressure-sensing floor, called MIDI Dance Floor, consisting of four strips of heavy duty plastic sheeting, each of size 4.87 m by 1.21 m and with 32 embedded FSR sensors of size 0.6 m by 0.6 m, for a total of 128 embedded FSR sensors [108]. It was used as an interface for musical instruments.

Paradiso *et al.* used a pressure-sensing floor as an interface for a musical instrument, measuring the foot pressure and position [105]. The floor was based on a 16x32 grid of piezo-electric wires hidden under a 3 m by 1.8 m carpet (see figure 2.5). It was coupled with a pair of Doppler radars to track the movement of the arms and upper body.

Schmidt *et al.* analysed the use of pressure-sensing surfaces in general, including floors, tables, and shelves [138]. Their sensing floor was monolithic, 2.4 m by 1.8 m in size, mounted on 4 load cells located in the corners (see figure 2.6). It could locate and track a single person in the environment, identify the addition and removal of single objects from the scene, as well as recognise when objects were being knocked over.

Figure 2.5 – The Magic Carpet presented by Paradiso *et al.* in [105].

(a) The floor in upright position.

(b) An enlarged view of the load cell supporting the floor.

Figure 2.6 – Monolithic sensing floor presented by Schmidt *et al.* in [138].

Modular pressure-sensing floors

Addlesee *et al.* developed the ORL active floor [3], which is a modular sensing floor made of tiles with load cells in the corners. Each load cell supports the corners of four adjacent load tiles (or two and one tiles for the sides and the corners of the floor, respectively), as shown in Fig. 2.7. The tiles are 0.5 m x 0.5 m in size, rigid, being made of 18 mm thick plywood with a 3 mm mild steel plate on top. The data sampling frequency was 500 Hz. The authors then used Hidden Markov Models (HMMs) to classify the detected footstep traces, in order to recognise the walking persons.

Orr and Abowd used a prototype composed of a single tile, 50 cm x 50 cm in size, made of a ≈ 1 cm thick steel plate, placed on 4 load cells, located in the corners (see Fig. 2.8). They recognised users by creating their footstep models, and then employing the *Nearest-Neighbor* algorithm in a multi-dimensional space for user identification [104]. The user models were based on their footstep profile features, such as the maximal load value during heel strike and during toe push-off, and the minimal load value recorded during the weight transfer from heel to toe.

Figure 2.7 – A load cell supporting two tiles of the ORL active floor. Presented by Addlesee *et al.* in [3].

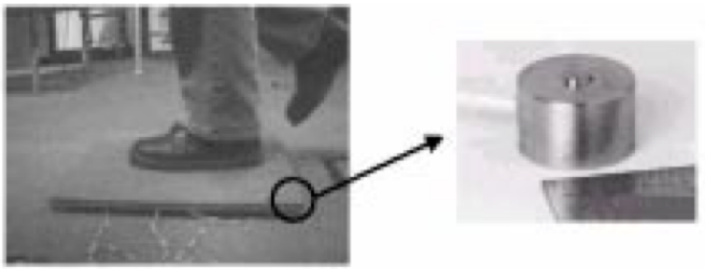


Figure 2.8 – Sensing floor plate and load cell presented by Orr and Abowd in [104].

Pirttikangas *et al.* from the University of Oulu developed a pressure-sensitive floor as part of a smart living room [110]. The 100 m^2 floor was made of stripes of EMFi material (30 vertical and 34 horizontal stripes, each 30 cm wide), which make up a 30x34 matrix of cells of size 30 cm x 30 cm, with a sampling rate of 100 Hz (see figure 2.9). It was used for the automatic recognition of occupants using the pressure pattern of their gait and discrete HMMs [110], Learning Vector Quantization (LVQ) [109], Distinction-sensitive Learning Vector Quantization (DSLVQ) [152], and multiple classifiers [150, 153].

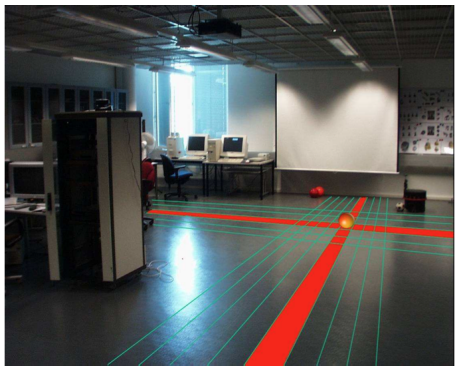


Figure 2.9 – The sensing floor composed of EMFi stripes, presented by Pirttikangas *et al.* in [110].

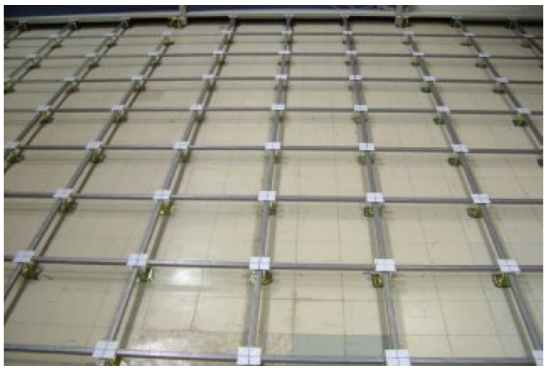


Figure 2.10 – The structure of the FSR floor presented by Lee *et al.* in [81].

Leikas *et al.* developed an immersive virtual environment containing a modular pressure-sensing floor [82]. The floor was made of 49 pressure-sensitive tiles, each having 4 sensors (one in each corner) of unspecified type.

Jung *et al.* tracked and recognised people walking on a *TechStorm foot analyzer* mat-type pressure sensor (TechStorm Inc., Korea) using HMM and Levemberg-Marquart learning [65], as

well as HMM and a Neural Network [66]. The size of the sensor was 0.8 m by 0.4 m, containing 80x40 sensors with a sampling frequency of 30 Hz, as shown in figure 2.11.

Figure 2.11 – The *TechnStorm foot analyzer* pressure-sensing mat, employed by Jung *et al.* in [66].

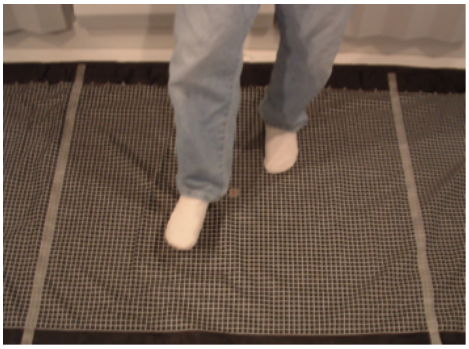


Figure 2.12 – The Xsensor pressure sensing pad used by Yin and Pai in [190].

Yin and Pai presented an intuitive animation interface, FootSee, that uses a pressure sensor pad, through which users control avatars for video games [190]. The sensor employed was a XSensor pressure pad [187], of size 2 m by 0.8 m, containing a 160x64 grid of pressure sensors as shown in figure 2.12, with a data sampling frequency of 6 Hz. The authors used an offline training phase to map full body motions to corresponding foot-ground pressure distributions on the sensor pad, and then employed inverse kinematics to calculate the posture of the human from the sensing data provided by the floor. The employed features are the velocity of the center of pressure (COP), contact area of both feet, and Hu invariant moments.

Lee *et al.* [81] presented a FSR floor, 7.2 m x 6.6 m in size, subdivided into blocks of 0.6 m x 0.6 m, each supported by 4 sensors located in its corners (see figure 2.10). They explored its use for tracking humans and as an interface for human-computer interaction. For instance, a visual interface was projected onto the floor, with which the users could interact by tapping buttons, and by changing their location and speed on the floor.

Richardson *et al.* developed a modular floor, consisting of a tesselation of interlocking tiles [123]. Each tile consists of 20 hexagonal FSR sensors, arranged in a shape that allows them to be self-holding, as shown in figure 2.13. The tiles contained within all the circuitry. The deployment of these tiles in sensing environments, including the self-localisation of the tiles composing the floor, was described in [90].

Figure 2.13 – Several interconnected Z-Tiles, presented by Richardson *et al.* in [123].

Rangarajan *et al.* designed a pressure-sensing floor consisting of 96 networked pressure-

sensing mats, arranged in a rectangular matrix of 12 rows by 8 columns, covering a surface of $\sim 16\ m^2$, and with a sampling frequency of 43 Hz (see figure 2.14) [119, 120]. Each sensing mat is 0.48 m by 0.43 m in size, and is embedded with a 48x42 array of 2016 FSRs, resulting in a resolution of approximately 1 sensor per cm^2. Each FSR is 6mm by 6mm in size, leaving little space between the sensors of the grid. The sensing resolution of the floor was high enough to allow the direct detection of the heel and toes. This floor was later used by Qian *et al.* to identify humans based on features of their gait [115, 116].

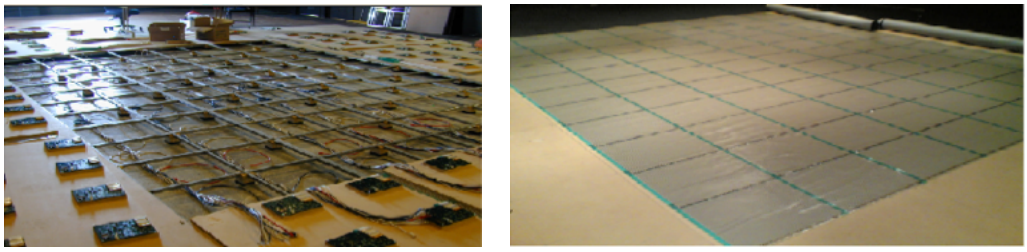


Figure 2.14 – The sensing floor developed by Rangarajan *et al.* in [119]

Wen-Hau *et al.* [182] developed a floor containing 25 sensing tiles of size 60 cm by 60 cm, with a single load sensor under each tile, and with a spacing of 20 cm between neighbouring tiles. This spacing left blanks in the sensing surface of the floor. To counter this, tracking of the inhabitants was performed using an algorithm called Probability Data Association (PDA), which is based on the Kalman filter. The inhabitants were tracked with an error of less than 28 cm between their estimated and real location.

Bose and Helal presented a tiled floor [23] for detecting humans and analysing their gait, developed for the Gator Tech Smart House [58]. Each tile of the floor had a piezoelectric force sensor embedded under the tile, attached to its central support (see figure 2.15).

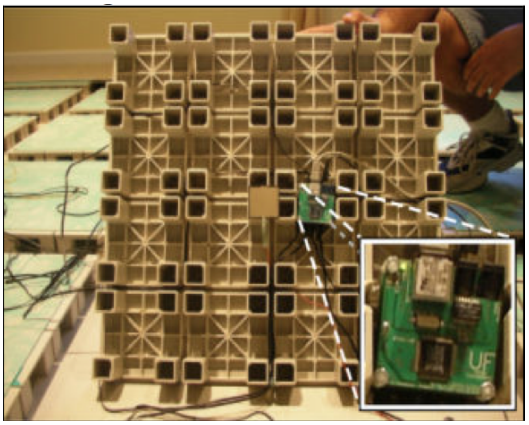


Figure 2.15 – Pressure-sensing tile presented by Bose and Helal in [23].

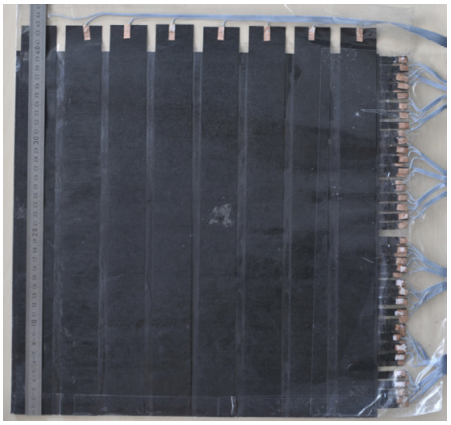


Figure 2.16 – The tacTiles presented by Anlauff *et al.* in [11].

Vera-Rodriguez *et al.* described a high-resolution pressure-sensing floor, which employs piezoelectric sensors mounted on a printed circuit board, and placed under a conventional mat [170]. Each mat was 0.3 m by 0.45 m in size, and contained 88 piezoelectric sensors. The floor was made of two such mats. The system extracted features from the footstep's ground reaction force (GRF) and it's footprint, and employed a Support Vector Machine (SVM) to recognise the users [172, 173, 170, 171].

Visell *et al.* used a tiled load-sensing floor as a human-computer interface [177, 179, 176], where an image of the interface is overlayed on the floor, on which users can press virtual buttons with their feet (see figure 2.17). The floor was composed of 36 tiles, each 0.3 m x 0.3 m x 2 cm in size, supported by 4 FSR sensors in the corners of the tile, as shown in figure 2.17a. The tiles were also equipped with vibro-tactile actuators beneath the plate, that could provide feedback to the users. This floor was used for 3D human posture tracking using Bayesian filters [118], and for providing the illusion of walking on materials such as gravel, snow, and sand [178].

(a) The locations of sensors under a tile.

(b) The floor pressure-sensing interface.

(c) A user interacting with floor-based interface widgets.

Figure 2.17 – The sensing floor presented by Visell *et al.* in [177].

Chang *et al.* introduced an interactive floor [32] with a total surface of 2.6 m by 2.2 m, composed of a 4x2 array of screens, each of size 1.096 m by 0.64 m. All these screens were supported by frames with 4 piezoresistive pressure sensors (one in each corner) for tracking the users' footsteps on the floor, as shown in figure 2.18. User identification was performed using an RFID system.

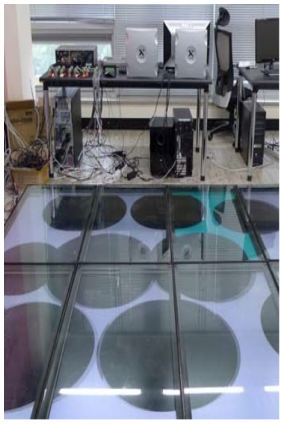


Figure 2.18 – The interactive floor presented by Chang *et al.* in [32].

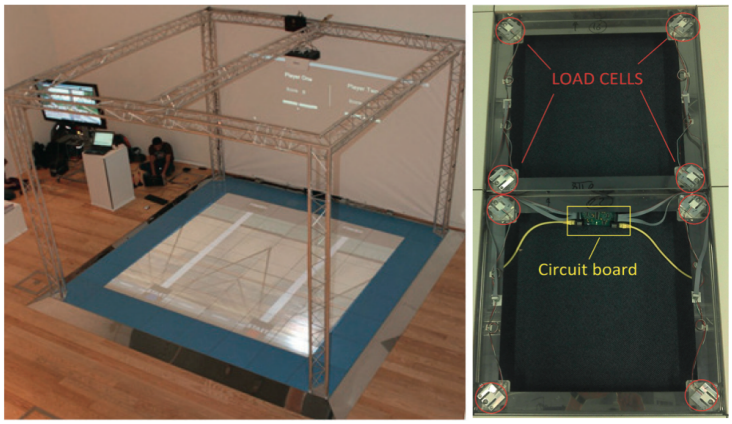


Figure 2.19 – The sensing floor presented by Heller *et al.* in [59]

Anlauff *et al.* developed pressure-sensing tiles, called tacTiles, using paper-based FSR [11]. Each tile was 0.4 m by 0.4 m in size, and incorporated a total of 64 FSR sensors placed in a 8x8 grid (see figure 2.16).

Klack *et al.* developed a tiled sensing floor [73], measuring 20 m^2 and composed of 64 wooden tiles (0.6 m by 0.6 m by 40mm in size), mounted on a solid steel frame to allow free space for the wiring and devices underneath the floor (see Fig. 2.20). The tiles were equipped with piezoelectric

sensors in the corners. It was used to detect characteristic walking patterns, fall events and other abnormal movement behaviours that would suggest an emergency situation [83].

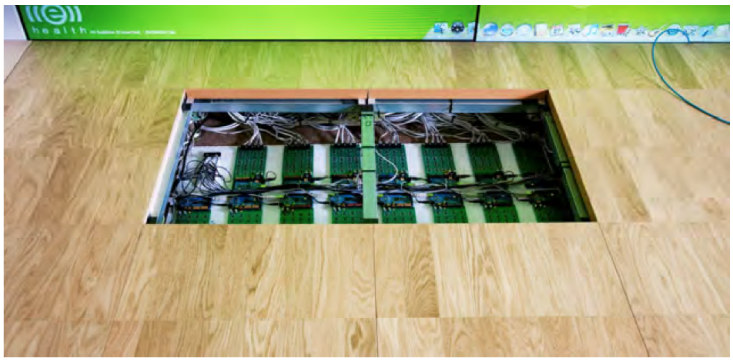
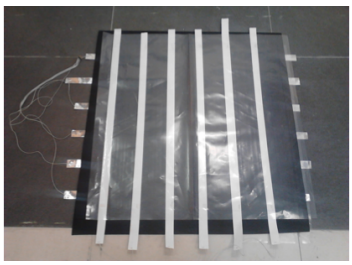


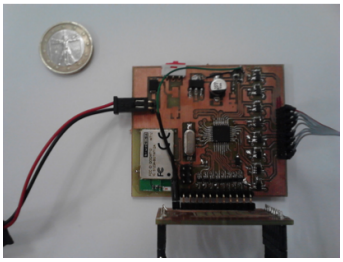
Figure 2.20 – The interactive floor presented by Klack *et al.* in [73].

Lombardi *et al.* [85] developed a tiled sensing floor, where each tile contains sensing stripes (see figure 2.21). The tiles are 0.6 m x 0.6 m large and 4.5 mm thick, and are made of ceramic. A sandwich sensing structure is placed under the ceramic tile: a conductive polymer is put between aluminium stripe electrodes. When a pressure is applied on top of the tile, the conductive rubber is compressed between the electrodes surface, increasing the contact area between the rubber and the electrodes, and decreasing proportionally the resistance between them. This resistance is acquired by a capturing board, which interprets it as a pressure value. The thin tile is flexible enough to avoid distributing the pressure equally over the sensors, allowing to recognise the shape of a foot placed on the tile. A hierarchical communication network connects all the tiles composing the floor. The authors were inspired by the computer vision community in their development of spatio-temporal techniques descriptors for the recognition of human behaviour. They used Randomised Tree classifiers to discriminate between the presence of an object and a person on a tile, as well as between human postures (standing, walking, jumping, lying), based on features that do not depend on the position of a person within a tile, like the matrix of pressures sensed by the tile, the mean pressure value and its variance over a temporal interval, and the movement of the barycenter.

(a) The installed floor prototype



(b) The contact-sensing stripes



(c) The capturing board

Figure 2.21 – The sensing floor presented by Lombardi *et al.* in [85].

Al-Naimi *et al.* presented a sensing environment combining a tiled pressure-sensing floor with a ceiling with PIR sensors, that was employed for the detection and tracking of humans. [4]. The floor was made of plywood tiles of size 0.5 m by 0.5 m by 0.022 m, each with 16 FSR sensors placed in a 4x4 array layout onto the plywood, and covered by a 6 mm medium-density fibreboard (MDF).

Table 2.2 – Features and techniques for human recognition using load-sensing floors

Authors	**Publ. Year**	**Floor sensor**	**Major features**	**Classifier**
Addlesee *et al.* [3]	1997	strain gauge load cells	Pressure profile over a footstep	HMM
Orr and Abowd [104]	2000	strain gauge load cells	Key points from pressure profile	K-nearest neighbors (KNN)
Schmidt *et al.* [138]	2002	strain gauge load cells	Weight	None
Pirttikangas *et al.* [110]	2003	EMFi	Pressure profile over the entire floor during walking	HMM
Pirttikangas *et al.* [109]	2003	EMFi	Pressure profile over the entire floor during walking	Learning vector quantization (LVQ)
Yun *et al.* [192]	2003	pressure switch sensor	Compensated foot centers over 5 consecutive footsteps	Multi-layer perceptron (MLP)
Jung *et al.* [65]	2003	pressure mats	2D trajectories of COP	HMM
Yin and Pai [190]	2003	pressure mat	COP velocity, feet contact area, Hu invariant moments	Inverse Kinematics
Jung *et al.* [66]	2004	pressure mats	2D positional trajectories of COP	HMM, Neural Network (HMM-NN)
Suutala and Röning [152]	2004	EMFi	Features from spatial, frequency domain over a footstep	DSLVQ
Middleton *et al.* [91]	2005	FSR mats	Stride length, stride cadence, heel-to-toe ratio	standard distance metric of similarity
Yun *et al.* [194]	2005	photo interrupter sensors	Compensated foot centers and heel-strike and toe-off time over 5 consecutive footsteps	MLP
Suutala and Roning [150]	2005	EMFi	Features from spatial, frequency domain over a footsteps	MLP, LVQ
Yun *et al.* [195]	2008	photo interrupter sensors	The footprint pattern and the sampled transitional footprints over combinations of 2 or 4 footsteps	MLP
Suutala and Röning [153]	2008	EMFi	Pressure and time features extracted from pressure profile over a footstep	MLP, SVM
Suutala *et al.* [154]	2008	on/off switch sensors	Single footstep: length, width, duration, number of pixels in the binary map, (min, max, mean, std) from the gray-level duration map; Between footsteps: stride, length, stride cadence	Gaussian Process
Qian *et al.* [115]	2008	FSR mats	2D trajectories of the Center of pressure, Pressure profile over time	Fisher linear discriminant (FLD)
Vera-Rodriguez *et al.* [173]	2009	piezoelectric force sensors	Geometric and holistic footstep data	SVM
Qian *et al.* [116]	2010	FSR mats	Mean pressure, stride length	FLD
Vera-Rodriguez *et al.* [170]	2010	piezoelectric sensor mat	Holistic pressure-time info	SVM
Yun [193]	2011	photo interrupter sensors	Foot centers, heel-to-toe time, footprint geometric data	MLP
Vera-Rodriguez *et al.* [171]	2013	piezoelectric sensor mat	Fusion of time and holistic pressure info	SVM
Inria SmartTiles [9]	**2015**	**strain gauge load cells**	**Weight over time**	**Knapsack algorithm**

Heller *et al.* developed a modular sensing floor of size 3 m x 3 m, consisting of 36 interconnected force-sensing tiles [59], each having the dimensions 0.5 m x 0.5 m, with strain gauge single axis load cells in each of the 4 corners (see Fig. 2.19). A short throw projector was used to project a visual interface on the surface of the floor. The floor was developed to study and train the dynamic balance of athletes. Interactive applications were also developed to fight child obesity, and to help rehabilitate elderly people for preventing falls.

Table 2.2 presents the floors on which human recognition experiments were made, together with the features extracted and the methods employed for recognition.

2.4.3 Other sensing floors

Sensing floors have been developed, that perceive the environment in ways different from pressure sensing. Valtonen *et al.* presented a 2D human positioning and tracking system, called TileTrack [169], which used a low-frequency electric field to locate humans on the floor. The system could only detect conductive objects, and did not provide information about the weight of objects. Braun *et al.* introduced a floor with capacitive sensors, called CapFloor [24], that can localise humans using low-intensity electric fields, and which was also used for fall detection. Ropponen *et al.* used a floor with a low-frequency RFID location system, based on a matrix of quad antennae under the floor surface [130, 131]. This floor was used for tracking people [124], and for detecting human falls [125].

2.4.4 Commercially available sensing floors

Some sensing floors are already commercially available today. Products like the SensFloor [80, 146] (a floor network of capacitive proximity sensors, see Fig. 2.25), and FloorInMotion [156] by Tarkett are being commercialised by companies mainly for the senior care industry.

Figure 2.22 – Kistler portable multi-component force plate for 10 kN, Type 9286B [72]

Figure 2.23 – The GAITRite Surface pressure-sensing floor [50]

Load-sensing surfaces are also employed in biomechanical and medical laboratories. Examples include the GAITRite gait analysis system [50], which is an electronic walkway (see figure 2.23), the Tekscan force measurement and tactile sensors [159, 87], the *TechStorm foot scan* pressure-sensing mat [157] (see figure 2.11), the XSensor pressure pads [187] (see figure 2.12), and the Kistler force plates [72] (see figure 2.22), which are used for sports and performance diagnostics, as well as for gait and balance analysis. The Active Gaming company proposed the pressure sensitive Lightspace Floor [160], which is an interactive gaming platform combining pressure sensors with LEDs for visual feedback, as shown in figure 2.24.

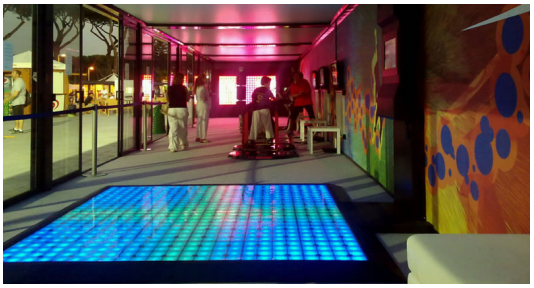


Figure 2.24 – The Lightspace Floor presented by the Active Gaming company [160].

Figure 2.25 – The SensFloor, presented by Lauterbach *et al.* in [80].

2.5 Conclusion

The domain of *sensing floors* has been in rapid expansion in the last twenty years, with applications evolving from entertainment to health monitoring and surveillance. The concept of force-sensing plates, with a non-negligible cost and only available in the doctor's office, is now being transformed into a vision where low-cost sensors cover the entire floor of a living environment.

Modular floors became more widespread, due to easier data intepretation and maintenance, and despite their more difficult installation as compared to sensing carpets. For a complete representation of the pressure exerted on the floor, and to make use of physical properties like mass conservation inside a scene with a constant composition of objects, the floor must perceive all the forces exerted on it. No sensing holes should be left inside the observed perimeter. Compared to flexible floor surfaces, where pressure sensors are required to cover the entire surface, rigid tiles allow to have fewer pressure sensors per unit of surface, by distributing the pressure to their supports.

Floors that perceive pressure information, as opposed to only presence detection, can be exploited for improved object tracking and recognition, by providing a way to check the hypotheses on the localisation of objects by verifying the presence of corresponding weights in their expected locations. Pressure-sensing floors also provide information for (multi-modal) activity recognition. Accelerometers can detect impacts with the floor (e.g., falls, ball bounces), although their location under the floor prevents them from detecting soft or low-intensity impacts.

Although existing floors have limited capabilities, being able to detect people lying on the ground and tracking those walking in the environment, they have the potential to become a versatile sensor for ambient intelligence. Invisible in their interaction with the users, they can provide space occupancy information for robot navigation, data on human localisation, and derivated analytics like activity monitoring, and health diagnosis for the continuous supervision of the well-being of persons. Today, sensing floor models with limited capacities are commercially available, mostly developed for the health care sector. However, much more can be done both in developing sensing and analytical capacities of floor sensors, and in their proliferation for usage in robotic systems.

This chapter has presented the existing sensing floor prototypes, including descriptions of their modularity, and of the sensors they use. The next chapter will introduce the sensing floor prototype designed by our research team at Inria, developed both for the perception and analysis of human activities, and for interaction with robots in their navigation and exploration tasks. This floor is used for all the applications presented in this thesis.

3

The Inria SmartTiles prototype

Contents

In the previous chapter we have surveyed the literature on existing sensing floor prototypes. This chapter will introduce the sensing floor prototype developed at Inria Nancy, which will be used throughout the rest of this thesis.

The Inria SmartTiles is a sensing floor prototype, conceived as a part of a bigger study of the habitat of the future. The floor is integrated into an artificial apartment (Fig. 3.1), where it provides spatially localised pressure sensing, memory and computing power, as it will be shown in the following sections.

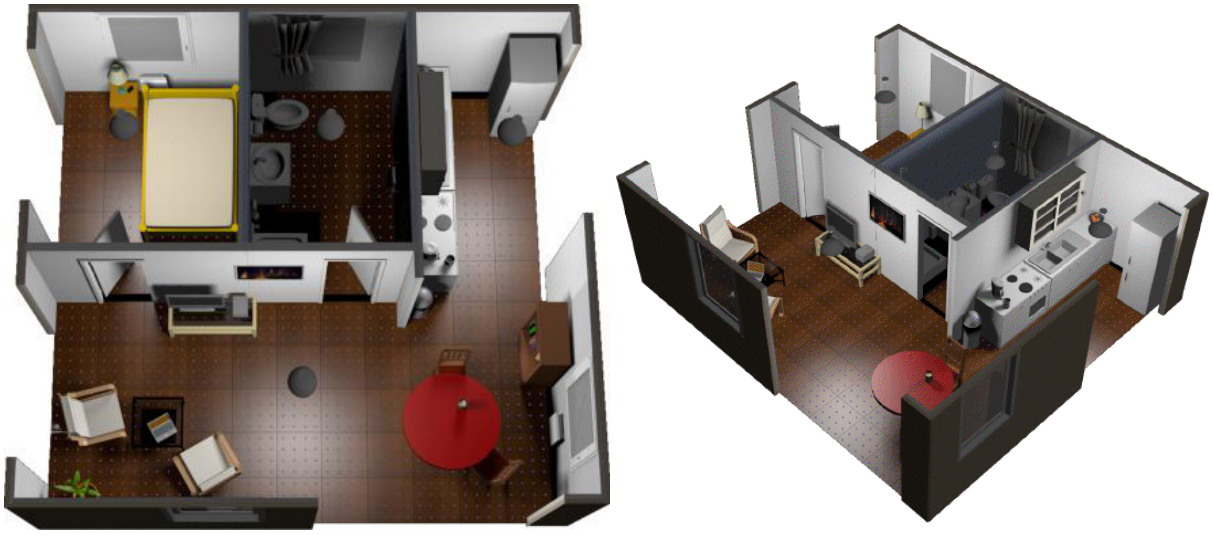


Figure 3.1 – A 3D model of the "Intelligent Apartment" prototype located at Inria Nancy

3.1 Origins of the project

This floor was developed to meet a double objective: (1) to serve as a pressure sensing floor for an ambient intelligence, and (2) to serve as a medium of interaction for bio-inspired robots. The floor is a discretized environment composed of tiles with embedded memory, capable of supporting bio-inspired navigation algorithms for mobile autonomous robots. Each tile composing the floor can store a limited amount of information, like the virtual pheromone trails laid by ant algorithms. This information can be later retrieved by other agents, or be used by the tiles to update their internal state, like in the case of pheromone propagation or evaporation. Robots evolving on the floor can peform tasks such as navigation, patrol, and exploration, using pheromone-based algorithms. The original idea of of a floor with communicating tiles for the implementation of environment-based multi-agent models of behavior was presented by Pepin *et al.* in [107].

The sensing floor was constructed according to the specifications drawn by leaders of the InfoSitu project, D.R. François Charpillet and Pr. Olivier Simonin, and engineers Dr. Olivier Rochel and Dr. Lionel Havet. The prototype was built by the company Hikob[3] and installed at its current location in the Inria Nancy Grand-Est research center at the end of 2012.

The project was funded by the Region Lorraine, the InfoSitu[4] project (standing for *Informatique Située*, i.e. *Spatially Localised Computing*), the Inria project-lab *Personally Assisted Living*[5], and the SATELOR project.

3.2 Architecture of the Inria SmartTiles prototype

The Inria SmartTiles prototype is a tiled sensing floor. The tiles are rigid, and have a size of 0.6 m by 0.6 m. Each tile can sense pressure, as well as memorise, compute and transmit data. Besides being equipped with a set of sensors, each tile also has an on-board processing unit, as well as a wireless and wired connection, which also provides electric power. These components are detailed in the following sections.

[3] http://www.hikob.com/
[4] InfoSitu `http://infositu.loria.fr/`
[5] Personally Assisted Living `https://pal.inria.fr/`

3.2.1 Processing units

Each tile is equipped with two ARM processors (Cortex m3 and a8), and a wired connection to the four neighbouring tiles, thus forming a sensor network (see Fig. 3.3). The wired RJ45 network connection also provides electric power. Both centralized and decentralized applications can be supported, thanks to the computing units embedded in the tiles, as shown in their architecture diagram (Fig. 3.2). The processing units were manufactured by Hikob.

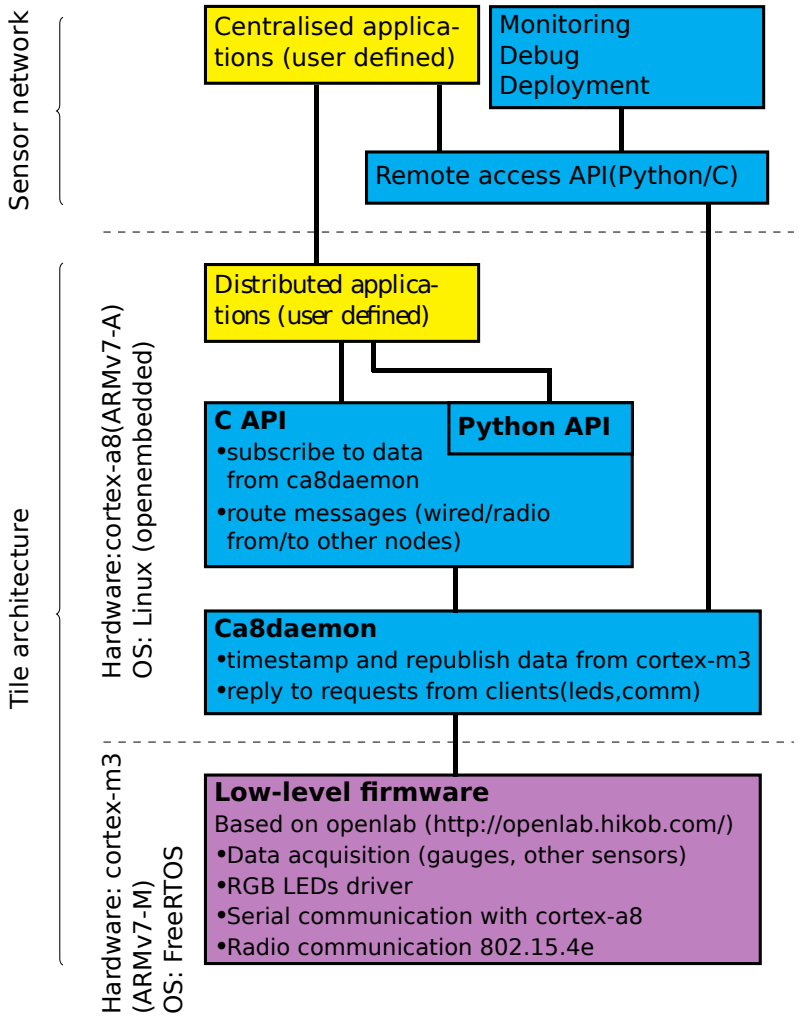
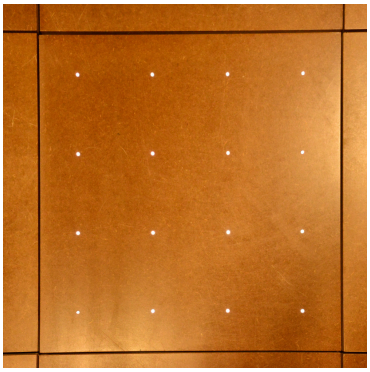


Figure 3.2 – Tile architecture. Low-level firmware is the violet block, blue blocks form the middleware, while high-level software blocks are in yellow.

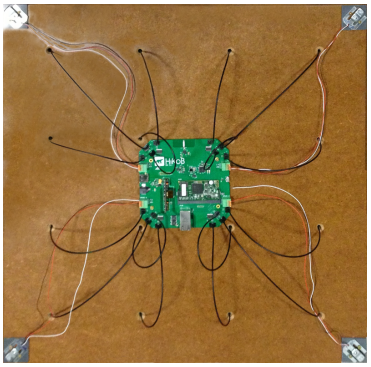
3.2.2 Communication

Communication between the tiles

The tiles are wired in a grid, similar to a cellullar automaton. This allows to use neighbourhood relationships between tiles for algorithms that employ stigmergy, like the diffusion of virtual pheromone traces. It can also support wavefront propagation algorithms for the exploration of unknown environments [16].



(a) Top view



(b) Bottom view

(c) Tiles in their context of use

Figure 3.3 – The load-sensing tiles. The bottom view shows the sensors located in the corners, as well as the CPU visible in the center. The black cables feed 16 light emitting diodes (LEDs) that are used for providing visual feedback.

Communication with the surrounding ambient intelligence

In our ambient intelligence architecture, all the sensor data is aggregated and centralized, in order to allow high-level reasoning tasks. The various nodes that we employ in our ambient intelligence tasks use Robot Operating System (ROS) [6] for intercommunication.

Communication with the mobile robots on the floor

For general purposes, the tiles can also communicate with other entities through either wired or wireless communication. This allows them, for instance, to provide navigation aid for mobile robots, as detailed in Chapter 7.

Figure 3.4 – PeKeeII robot communicating with an interactive sensing floor

3.2.3 Sensors embedded in the tile

Each tile is equipped with:

- 4 pressure sensors located at the corners of the tile;

[6] http://www.ros.org/

- 1 accelerometer in the center of the tile;
- 1 magnetometer.

The pressure sensors employed (strain gauge load cells) measure the load forces exterted on the floor. The embedded accelerometers detect shocks, that can be caused by objects or humans falling on the ground. The magnetometers serve to detect metallic masses located on the tiles, such as robots. Each tile also has 16 light-emitting diodes which provide visual feedback. The sensors are queried periodically for measurement data, with a frequency of 50 Hz. The format of the data sent by each tile is as follows:

Sensor type	**Sender IP**	**msg ID**	**Timestamp**	**Load Data (s_1 s_2 s_3 s_4)**
Gauge	192.168.1.24	7090	1378368585070	1639 1973 1690 2092

Table 3.1 – Format of data package containing pressure measurements.

Sensor type	**Sender IP address**	**msg ID**	**Timestamp**	**Acceleration (x y z)**	**Magnetic field (x y z)**	**Temp**
Lsm	192.168.1.24	7091	1378368585080	-26 25 -1059	8 -11 92	1696

Table 3.2 – Format of a data package containing acceleration, magnetic field, and temperature measurements.

The data packet contains two lines of information: one for the load cells, and one for the combined data of the accelerometer, magnetometer, and thermometer. The header of each message includes the type of the sensor providing the data, the IP address of the tile sending these data, the identification number of the message, and the Unix time in milliseconds of recording. The *Gauge* line contains the force values measured by the 4 pressure sensors, in their own conventional units. The *Lsm* line contains the acceleration values for 3 axes, the magnetic field values for 3 axes, as well as the temperature sensed by the tile.

My work was concentrated on the pressure sensors, which are presented in more detail below.

3.2.4 Specification of the load sensor

The Inria SmartTiles employ load sensors of the brand *SparkFun SEN-10245*, illustrated in Fig. 3.5). The specification of this sensor is shown in Fig. 3.6.

3.2.5 Load measurement linearity

The function that converts the conventional pressure units measured by the sensor to an equivalent value in newtons has a linear shape for the employed load sensors. This was checked by recording the pressure measurements for a series of known weights placed at rest on the center of the tile. These measurements showed a linear correspondence between the measured and real values (see Fig. 3.7), with a slope of 7.49 (i.e. a static load of 1 kg corresponds to an increment of 7.49 conventional sensor units).

3.2.6 Measurement noise

The load sensors of the tile are subjected to noise, as visible in Fig. 3.8a. The measurement error of individual sensors is comprised between ±1.25 kg, with the combined measurement error

Figure 3.5 – Images of the *SparkFun SEN-10245* load sensor. Source: `https://www.sparkfun.com/products/10245`

Capacity	kg	40-50
Comprehensive Error	mv/v	0.05
Output Sensitivity	mv/v	1.0±0.1
Nonlinearity	%FS	0.03
Repeatability	%FS	0.03
Hysteresis	%FS	0.03
Creep	(3min)%FS	0.03
Zero Drift	(1min)%FS	0.03
Temp. Effect on Zero	%FS/10℃	1
Temp. Effect on Output	%FS/10℃	0.05
Zero Output	mV/V	±0.1
Input Resistance	Ω	1000±20
Output Resistance	Ω	1000±20
Insulation Resistance	MΩ	≥5000
Excitation Voltage	V	≤10
Operation Temp. Range	℃	0--+50
Overload Capacity	%FS	150

Figure 3.6 – Specification of the *SparkFun SEN-10245* load sensor. Source: `https://www.sparkfun.com/products/10245`

typically oscillating between ±2 kg, as seen in figures 3.8 and 3.9.

The frequency distribution of the noise registered by the sensors is non-Gaussian, as seen in Fig. 3.9a. However, when summing the pressure values measured by the sensors of a tile (which are finite-variance random variables), the obtained frequency distribution of the total pressure intensity resembles a Gaussian, as suggested by the central limit theorem (see Fig. 3.9b).

The measurement error influences the localisation precision of forces exerted on the ground, as seen in the next section.

3.2.7 Localisation error

Localisation of static objects

The floor can locate the exerted punctual pressures, with a higher accuracy than the size of a tile. Punctiform pressures can be located through a calculation of the center of pressures measured by the load sensors, using the formula:

$$\text{Center of Pressure} = \frac{1}{\sum_i^{\text{total sensors}} |\vec{R}_i|} \sum_i^{\text{total sensors}} \left(\vec{R}_i \times coords(Sensor_i) \right) \tag{3.1}$$

where $\vec{R}_i$ is the reaction force measured by the i^{th} sensor supporting the tile, and the "$\times$" symbol is the *vector product* operator. Equation 3.1 is obtained from the static equilibrium equations, as further detailed in section 4.4.3.

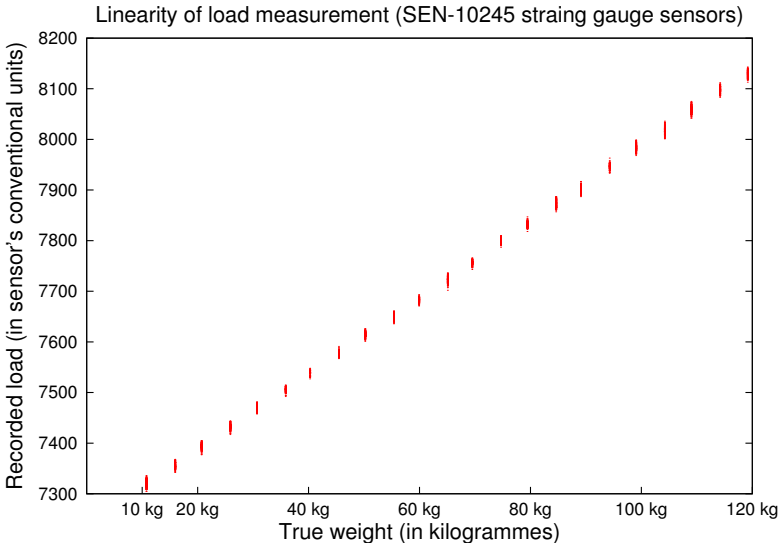


Figure 3.7 – Load measure linearity for a tile with 4 *SparkFun SEN-10245* load sensors. The intervals that can be seen instead of expected dots are due to measurement noise. The calculated slope of the line is 7.49 conventional units for 1 kg of static load.

In Fig. 3.10, we show the localisation precision for a vertically standing barbell, with weight disks screwed onto it. The localisation precision is influenced by the signal/noise ratio: a 7 kg punctiform load can be located with a ±10 cm precision, while a 20 kg load can be located with a precision of ±4 cm.

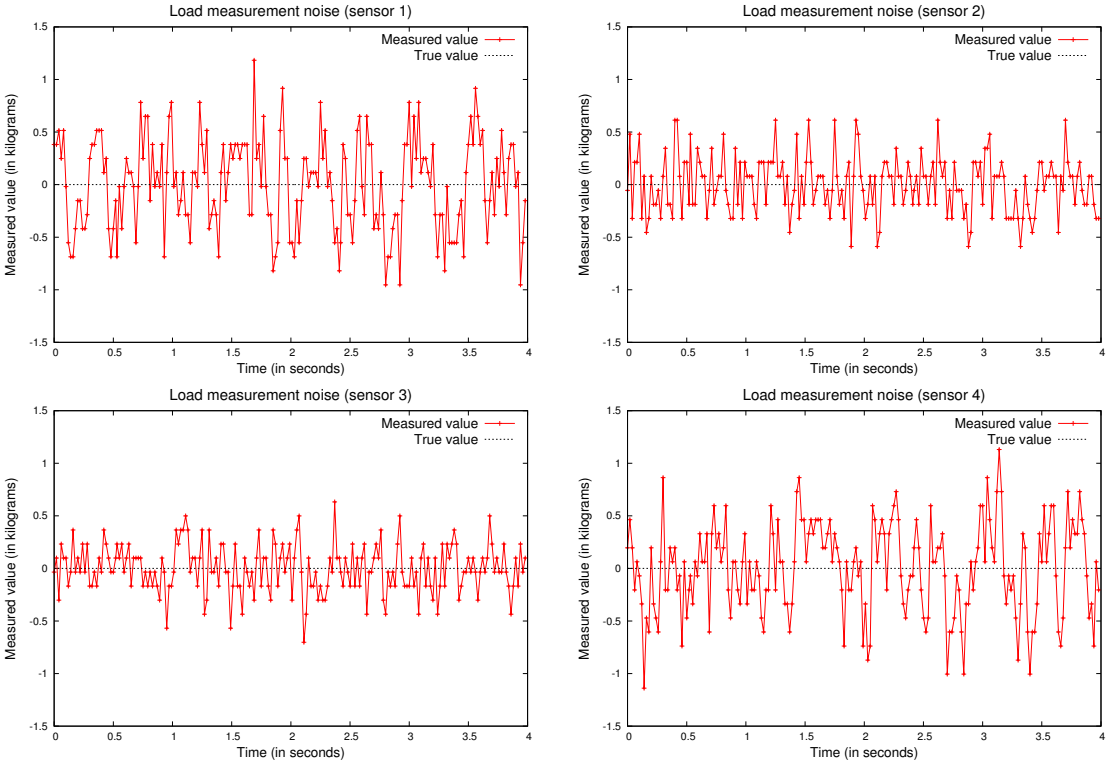
Localisation of moving objects

The tiles can also locate moving objects. To gain an understanding of the baseline precision of the floor sensor, we performed an experiment with a non-holonomic 4-wheeled robot (robuLAB-10 by Robosoft[7]) weighing 35.7 kg, rolling on the floor of the apartment (see Fig. 3.11). The idea was to track the fluid movement of an autonomous robot, as compared to the saccadic movements of the COP which are characteristic for the human gait.

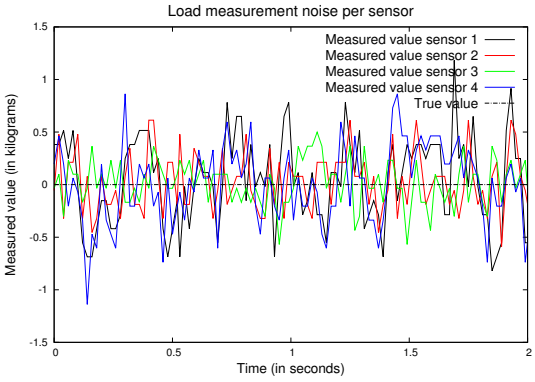
As we had no ground truth for the localisation of robot's center of pressure, we used an approximation using the data from a Qualisys [8] motion tracking system. Given that the robot is rigid, we could estimate the position of its COP using the least squares method, by calculating the point which minimized the quadratic distance error between itself and the center of pressure calculated by the sensing floor. The localisation precision for different types of trajectories is presented in Fig. 3.12.

We performed the same experiments with a lighter Turtlebot 2 robot, weighing only 6.3 kg (see Fig. 3.11c). The robot trajectories included: statically standing on the spot, making a rotation on the spot, riding in a straight line, in a rectangle-shaped trajectory, and in a figure eight trajectory. These trajectories were recorded in a rectangular area covered by a 3 by 5 grid of sensing tiles. The localisation was performed using three different methods: a direct estimation of the center of pressure using all the pressure forces sensed by the tiles that detected a non-zero load (DE-TS, standing for Direct Estimation with Tile Selection); a Kalman filter (KF) that

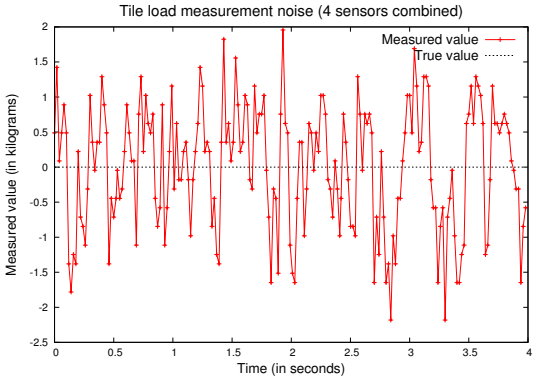
[7] http://www.robosoft.com/
[8] http://www.qualisys.com/



(a) Intensity of the noise measured on the 4 sensors of a tile, shown here in mass equivalent. The amount of noise varies across sensors, but is generally below $\pm 1.25\,\text{kg}$.

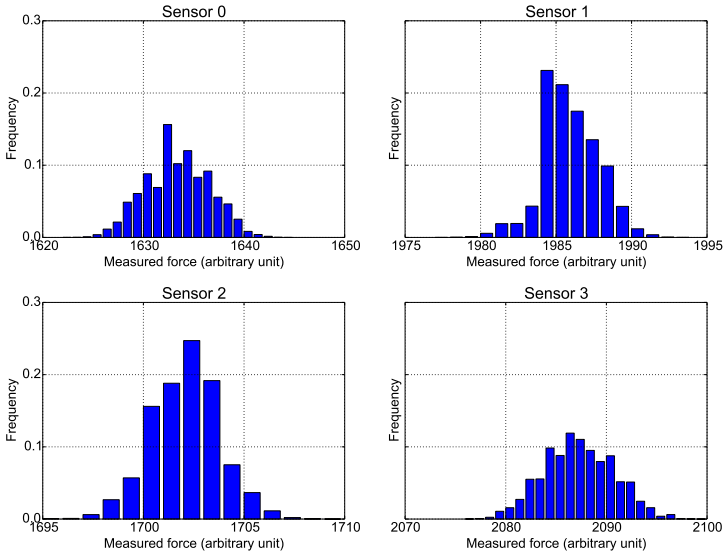


(b) The measurement noise values for the 4 sensors presented above in Fig. 3.8a, shown here together on a common scale.

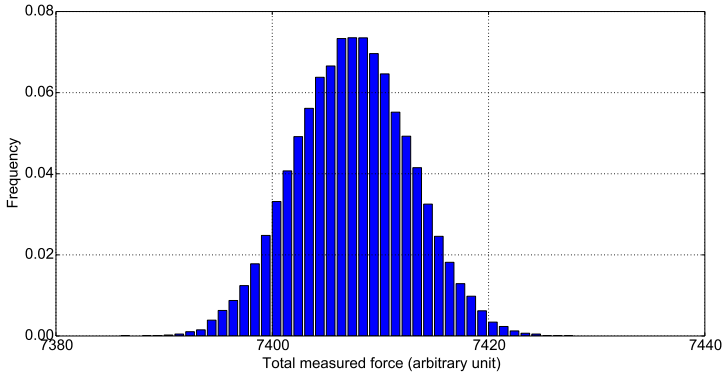


(c) Combined measurement noise for the 4 sensors presented above in Fig. 3.8a. Summing the measurement values over multiple sensors increases the intensity of noise, reaching here an equivalent of nearly $\pm 2\,\text{kg}$.

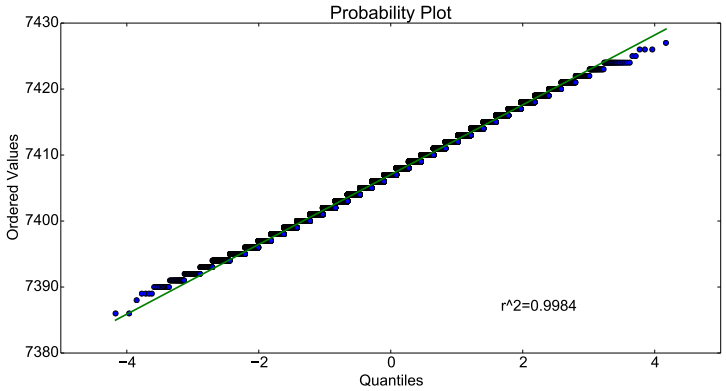
Figure 3.8 – Tile load measurement noise: noise intensity for individual sensors, and for all 4 sensors combined. Data obtained by recording the pressure data of a tile with no load on top. The weight of the tile itself, supported by the sensors, was 10.7 kg.



(a) The frequency distribution (histogram) of the pressure values measured by the sensors of a tile. Conventional pressure units (proper to the sensors) are shown on the horizontal axis, while the vertical axis indicates the frequency of the obtained measurement. Measured weight was 15 kg. The sensor has a randomly preset offset for zero pressure. 7.5 arbitrary units of pressure approximately correspond to 1 kg of load. The noise distribution is non-Gaussian.



(b) Frequency distribution (histogram) of the combined sensor measurement noise, based on the 4 sets of sensor measurements presented above in Fig. 3.9a. Due to the central limit theorem, the frequency distribution of the combined sensor noise measurements has approximately the shape of a normal distribution.



(c) Q-Q plot for the combined sensor measurements, comparing the experimental data to a normal distribution.

Figure 3.9 – Statistical analysis of the tile load-measurement noise.

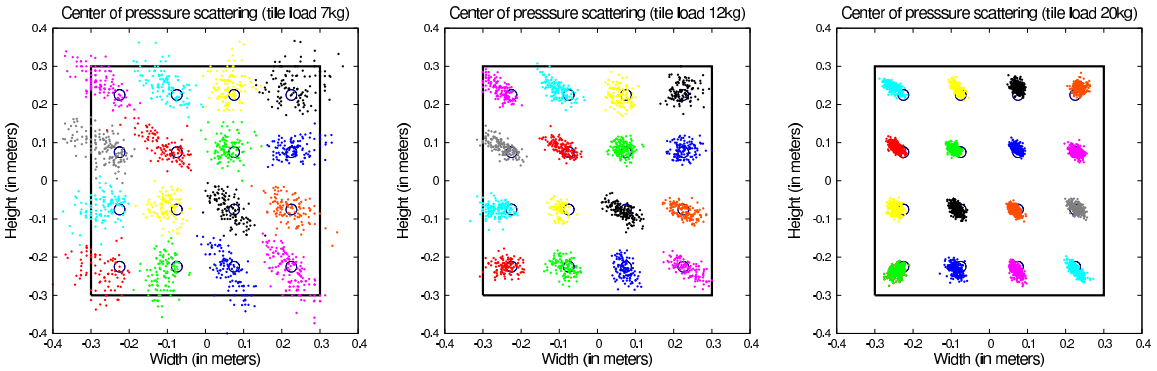


Figure 3.10 – The precision for the localisation of static objects. The weighed object was a vertically standing barbell, with weight disks screwed onto it. The thick black square represents the load-sensing tile. The scattering of the calculated center of pressure is caused by the sensor noise. Scattering is shown for 3 different loads (7 kg, 12 kg and 20 kg) at 16 different locations of the exerted punctual pressure, marked by black circles. The higher is the ratio of signal to noise, the less scattering is observed. The indicated load does not include the weight of the tile itself, which is 10.7 kg.

tracked the position of the robot through time from the noisy direct estimates; and an Extended Kalman filter (EKF) that also tracked the position of the robot and which incorporated the force value at time t into the state of the filter.

The localisation results, classified by trajectory type and localisation method are given in Table 3.3. The localisation error distributions are presented in Fig. 3.13. All the compared methods were capable of tracking the robots with an average error of approximately 6 cm. The KF outperformed the two other models, having 1 cm less in average error. This result is statistically significant at p-value ≤ 0.01, assessed by a bootstrap test for the difference between two sample means (2000 samples). A graphic comparison between the ground truth and the robots' trajectories computed by the floor is shown in Fig. 3.14 A co-authored paper on these probabilistic localisation techniques using a tiled sensing floor that details these results was submitted to the International Conference on Robotics and Automation (ICRA) 2016 [126].

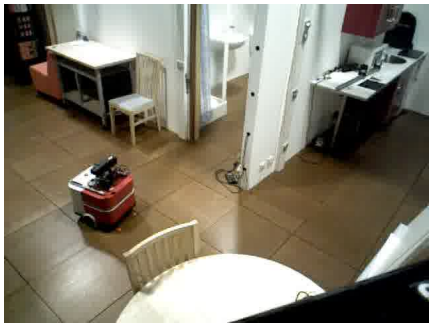
3.2.8 Synchronisation between tiles

The tiles send data packets at a frequency of 50 Hz. The tiles are synchronised using Network Time Protocol [92], with a resulting desynchronization of approximately 2 milliseconds among the 100 tiles composing the floor. This allows to assemble the data into pressure images of the entire floor, one image per 20 ms time-window.

3.3 Floor capabilities

The main emphasis of our research on this floor sensor is on its sensing capabilities for interoperating with an ambient intelligence, and its analytical capabilities for performing a continuous evaluation of the healt state of inhabitants, using the biomedical data that it can extract. Several functionalities have already been implemented on this prototype floor, including weight measurement, fall detection, extraction and tracking of footsteps, as well as entertainment games, which are detailed below. However, more research and engineering effort is required for integrating all

(a) A nonholonomic robuLAB-10 robot by RoboSoft, used in our experiment, weighing 35.7 kg. Source: www.robosoft.com



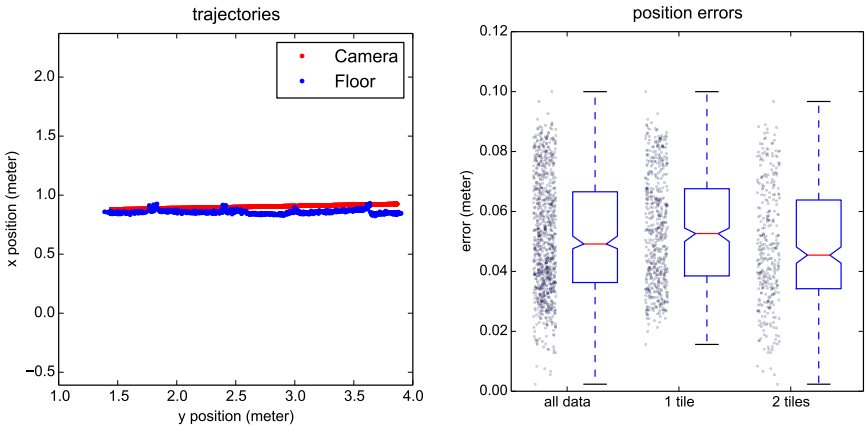
(b) A robuLAB-10 robot navigating on the sensing floor.

(c) A Turtlebot 2 robot, weighing 6.3 kg with the notebook computer. Source: www.turtlebot.com

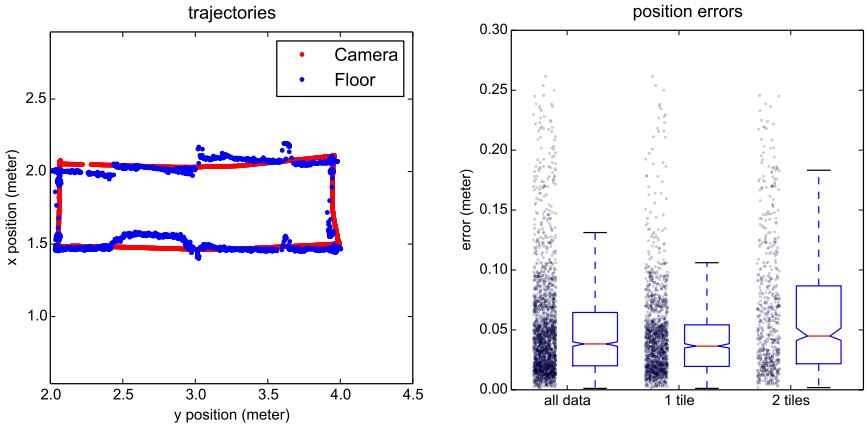
Figure 3.11 – Image from an experiment for calculating the precision of localisation for mobile objects rolling on the floor. A heavy RobuLAB-10 robot (35.7 kg) and a light Turtlebot 2 robot (6.3 kg) were used.

Turtlebot	Methods		
Trajectories	DE-TS	KF	EKF
all	7.1 cm (±6.4)	5.9 cm (±2.8)	6.7 cm (±6.1)
static	6.9 cm (±4.9)	6.2 cm (±2.4)	5.8 cm (±2.6)
rotation	5.8 cm (±7.4)	4.7 cm (±2.5)	4.4 cm (±2.1)
straight line	7.6 cm (±8.3)	6.0 cm (±2.6)	7.8 cm (±8.1)
rectangle	6.9 cm (±6.2)	5.7 cm (±2.9)	6.8 cm (±6.8)
figure eight	7.6 cm (±5.7)	6.2 cm (±2.7)	6.7 cm (±4.9)
Robulab	Methods		
Trajectories	DE-TS	KF	EKF
all	5.6 cm (±2.9)	4.6 cm (±2.7)	6.1 cm (±4.9)
static	1.5 cm (±1.1)	1.3 cm (±0.5)	1.4 cm (±0.8)
rotation	5.3 cm (±2.6)	5.3 cm (±2.5)	5.3 cm (±2.5)
straight line	6.1 cm (±1.5)	5.0 cm (±1.1)	6.7 cm (±3.8)
rectangle	5.9 cm (±3.3)	5.0 cm (±3.4)	7.3 cm (±6.5)
figure eight	5.8 cm (±2.4)	4.5 cm (±1.6)	5.4 cm (±2.2)

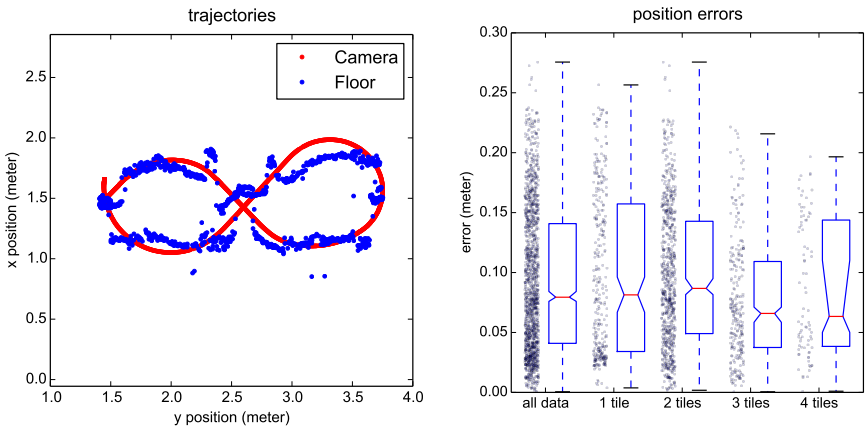
Table 3.3 – Summary values for the evaluation of the localisation methods, as mean errors (± sd.) in centimeters. Top: with the light robot. Bottom: with the heavy robot.



(a) Robot navigating in a straight line, being supported by 2 tiles at most at any given time.



(b) Robot following a rectangle-shaped trajectory, being supported by 2 tiles at most at any given time.



(c) Robot freely navigating on the floor, alternatively supported by 1-2-3-4 tiles.

Figure 3.12 – The precision for the localisation of a mobile object (a robuLAB-10 robot navigating on the floor). On the left, the robot trajectories (ground truth) are shown in red, while the localisation given by the floor is in blue. On the right, the Tukey box plots show the floor's localisation error, depending on the number of tiles supporting the robot at the time of localisation.

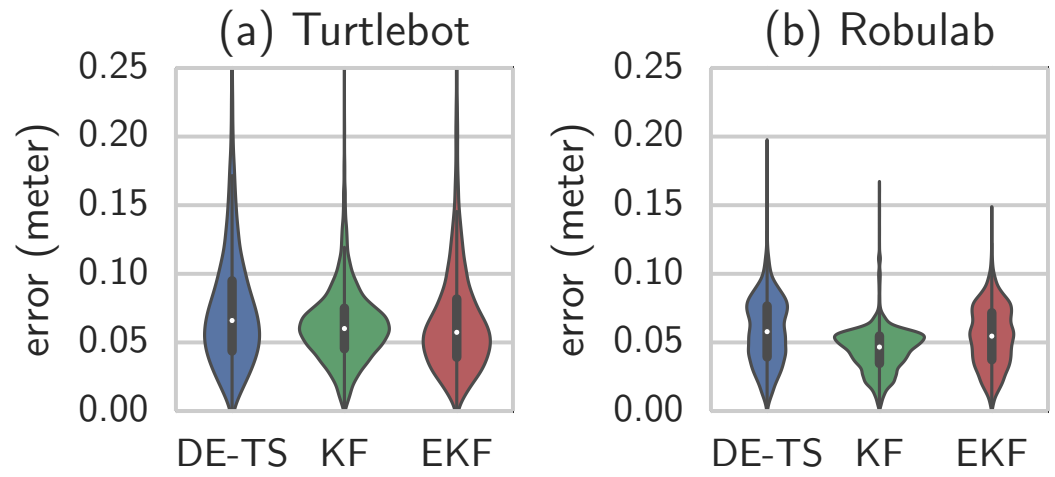
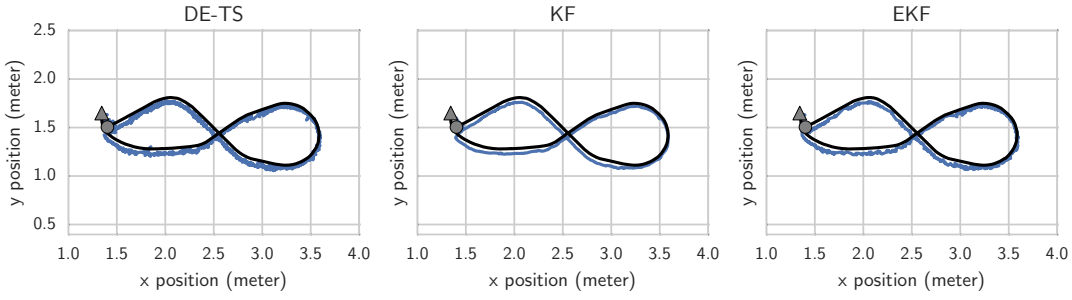
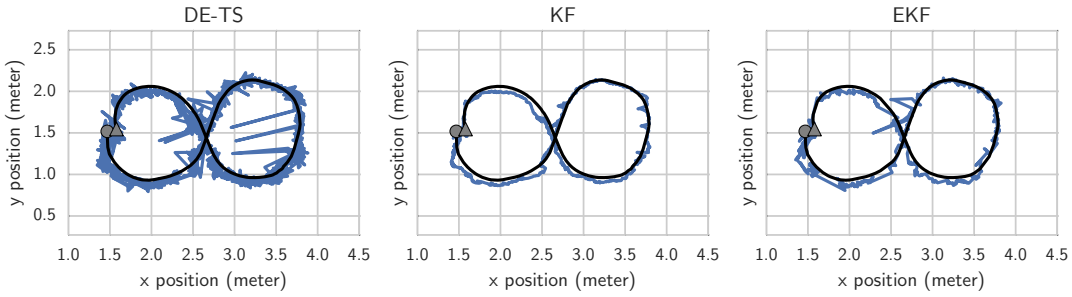


Figure 3.13 – Comparison of localisation error distributions for all models, in the figure eight scenario. Left: on the light robot; right: on the heavy robot.



(a) RobuLAB-10 robot, with a mass of 35.7 kg.



(b) Turtlebot 2 robot, with a mass of 6.3 kg.

Figure 3.14 – Estimated trajectories in a figure eight scenario. Black line: ground truth trajectory; blue line: estimated trajectory. Grey circle: starting point; grey triangle: stopping point. Top row: the heavy robot; bottom row: the light robot.

the functionalities into a single piece of software, which is expected to be based on high level reasoning.

3.3.1 Weighing scales

Each tile can be used as a weighing scale. The tiles provide feedback about the weight placed on them, using the LEDs and a color code (red light = 10 kg, yellow light = 5 kg, green light = 1 kg). The user can then sum up the corresponding values to obtain the final result (see Fig. 3.15).

Figure 3.15 – Using the load-sensing tiles as weighing scales. The LEDs encore different units of weight with different colors (red = 10 kg, yellow = 5 kg, green = 1 kg).

3.3.2 Extraction of footsteps

This floor is capable of detecting and measuring footsteps with high accuracy (see Fig. 3.16), extracting them using the variations in the translation speed of the center of pressure, as described by Ballaz *et al.* [13]. We also implemented real-time multi-user localisation (without user identification) based on the *Nearest Neighbor* heuristic using this prototype floor. This tracking and footstep-extraction capabilities open the door for the analysis of the human gait, as usually performed in the literature [185].

3.3.3 Evaluation of a person's frailty

In the medical domain, frailty is defined as a clinical syndrome, in which three or more of the following criteria are present: unintentional weight loss (5 kg in the past year), self-reported exhaustion, weakness (grip strength), slow walking speed, and low physical activity [49]. Among these 5 criteria, our sensing floor can detect 3: weight variation over time, changes in walking speed, and the overall physical activity. Therefore, it can be used as a tool for evaluating the frailty of a person.

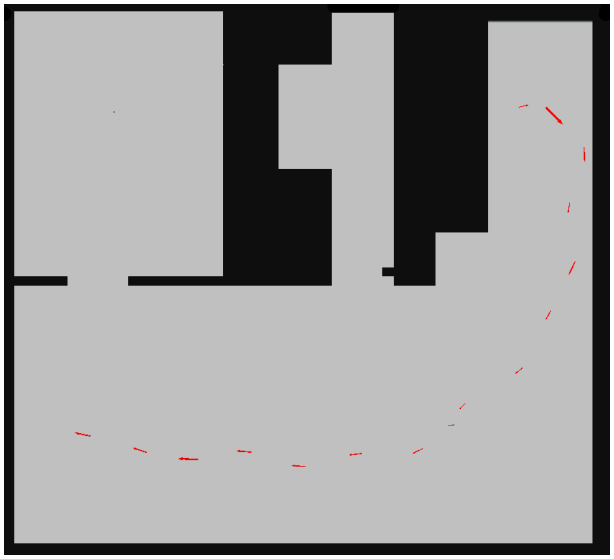


Figure 3.16 – Extraction of footsteps by tracking the evolution of the center of mass of a person using the load-sensing floor. The steps are overlayed on a 2d top view of our prototype apartment.

3.3.4 Fall detection

The tiles composing the floor have embedded accelerometers, which allow them to detect and localise hard falls, as shown in Fig. 3.17. However, the problem persists for soft falls, in which a person slowly collapses instead of suddenly hitting the ground. This makes soft falls difficult to detect with accelerometers embedded into the floor.

3.3.5 Visual guidance during night-time

The floor can calculate shortest paths through the unobstructed environment between the location of a person and the facilities usually used at night (e.g. bathroom, kitchen). These paths can then be displayed on the floor using the LEDs embedded into the tiles (see Fig. 3.18).

3.3.6 Tracking breathing when sleeping in bed

Load sensors placed under the bed can track the breathing of a person in order to detect anomalies such as sleep apnea, or simply to identify the presence of breathing [18]. They can also track the lying position of the person in bed [19], which may be helpful for training individuals to sleep in a particular position, as in the case of people with positional sleep apnea, or with gastroesophageal reflux disease.

3.3.7 Entertainment

The floor may also be used for entertainment. For instance, there are enough LEDs to serve as a screen for a low-resolution game like Tetris (Fig. 3.19). One of the tiles is used as a controller, which detects the position of the player's center of mass. It allows the player to move the incoming figures sideways by relying more on the left/right leg, to turn them by leaning forward on the toes, or to accelerate them by leaning back on the heels.

Figure 3.17 – The floor detects and localises hard falls using its accelerometers

3.4 Conclusion

In this chapter, we have presented the sensing floor prototype developed at Inria. Compared to the previous sensing floor prototypes, we have the additional capacity to store information and perform distributed computations on the tiles of this modular floor. The tiles being rigid, and supported by pressure sensors (in the corners of the tile), they are able to detect and measure pressure exerted in any place on the tile. Thus, rigid tiles demand a lower number of sensors per unit of surface, compared to traditional flexible sensing mats.

Due to this property, objects on the floor can be located with a resolution higher than the size of a tile, using a computation locating the center of pressure (COP) on the tile. This provides a cheap way to precisely locate objects on the floor. However, compared to high-resolution sensing mats, where footsteps can be detected by segmenting the corresponding footprints, other solutions have to be used for low-resolution tiled floors. We currently perform it by tracking the velocity of the COP on the ground, whose saccadic movement corresponds to the walking phases.

We also provided an overview of the preliminary applications already implemented in our laboratory setting. More advanced applications developed on this floor during this thesis will be presented later in Part II and Part III, together with directions for future work in Chapter 9.

In the next chapter, we will describe the software that we developed for processing and interpeting the raw information generated by our sensing floor.

Figure 3.18 – The floor providing visual guidance to the bathroom at night

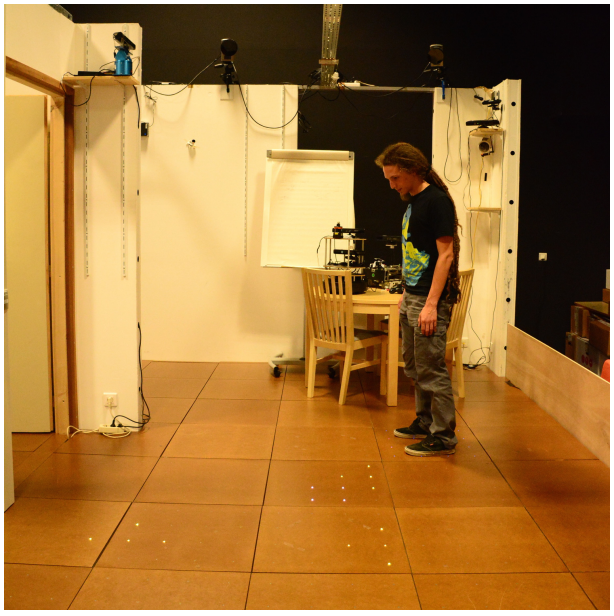


Figure 3.19 – Playing Tetris on the load-sensing floor.

4

Processing and simulating data from the load-sensing floor

Contents

In this chapter, we will introduce the software developed for working with our sensing floor prototype, presented in Chapter 3. The software was built for processing and visualising the data sent by the sensing floor. We also attempted to build a simulator, that would compute the distribution of exerted forces on the sensors supporting the tiles, considering them infintely rigid for simplicity. However, due to the number of sensors supporting a tile, we ran into difficulties with computing analytical solutions to this problem. These aspects are detailed in the sections below.

4.1 Architecture of the data processing software

The data processing software is composed of 3 types of functional units: the parser of the raw data incoming from the floor, the containers of the processed floor data, and the analytical units that segment, track, and locate objects on the floor. They are presented in different colors in the software architecture diagram, in Fig. 4.1.

The data parser sends the processed data to the container of the floor video, which groups the tile pressure data by their recording timestamps into snapshots of the entire floor. It uses prior knowledge on the positions of tiles and their sensors, which correspond to the ones shown in Fig. 4.2. The floor video container displays chronologically ordered 2D views of the floor,

with its occupancy state, measured pressure, and the located objects, as shown in Fig. 4.3. The analytical units process these floor snapshots, using centralised algorithms for segmenting, tracking and localising objects. Decentralising these algorithms and avoiding the data aggregation is a direction for future work. Finally, the software provides information on the localisation of objects, by publishing it to a ROS topic.

The software was written using the Java programming language, and it uses rosjava for its ROS communication. The program runs on a Linux Ubuntu 12.04 Precise Pangolin distribution, with ROS Hydro Medusa.

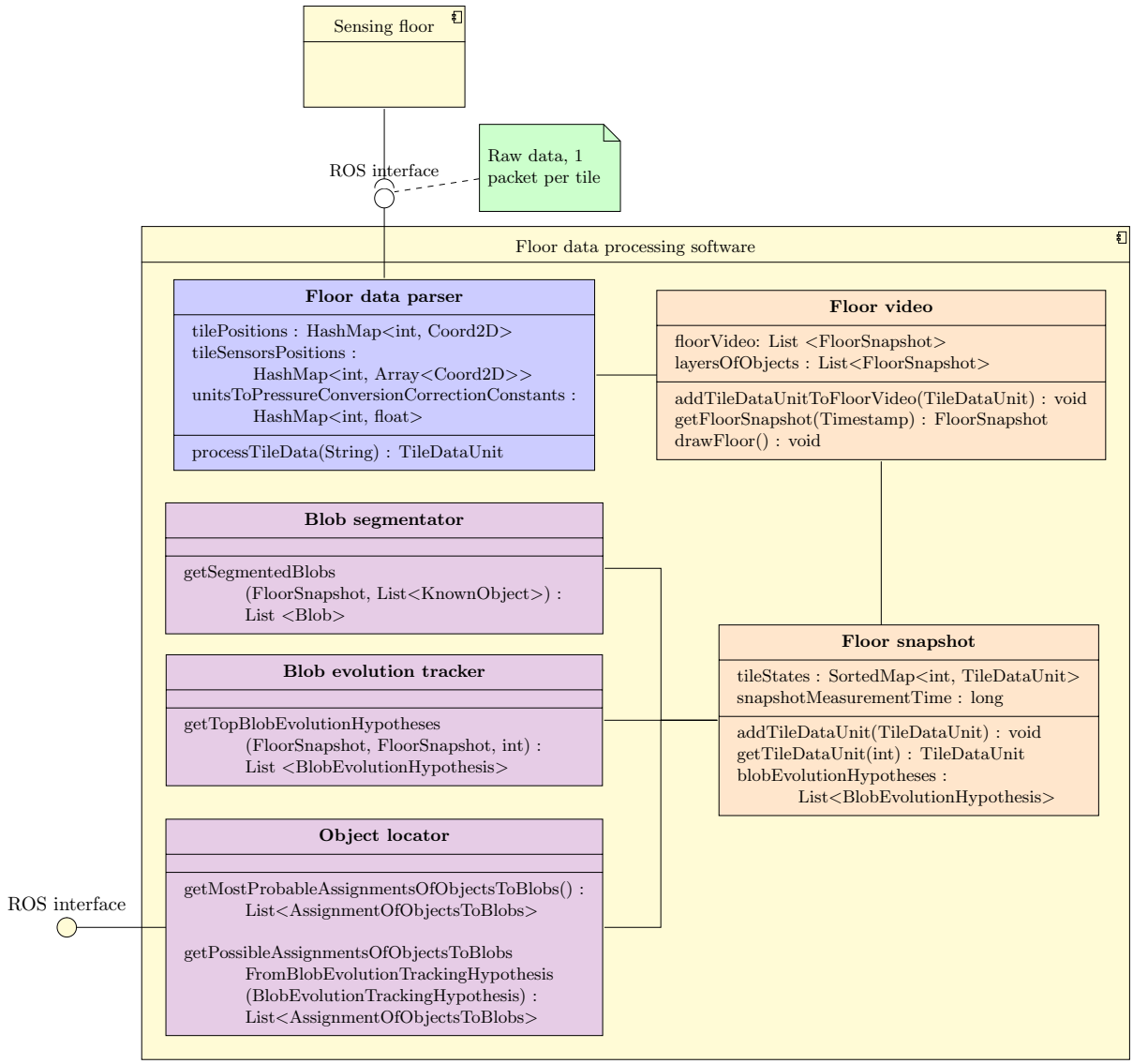


Figure 4.1 – The architecture of the floor data processing software. The data processing software is composed of 3 types of functional units: the parser of the raw data incoming from the floor (shown in blue), the containers of the processed floor data (orange), and the analytical units that segment, track, and locate objects on the floor (violet). For clarity reasons, only the important classes are shown here.

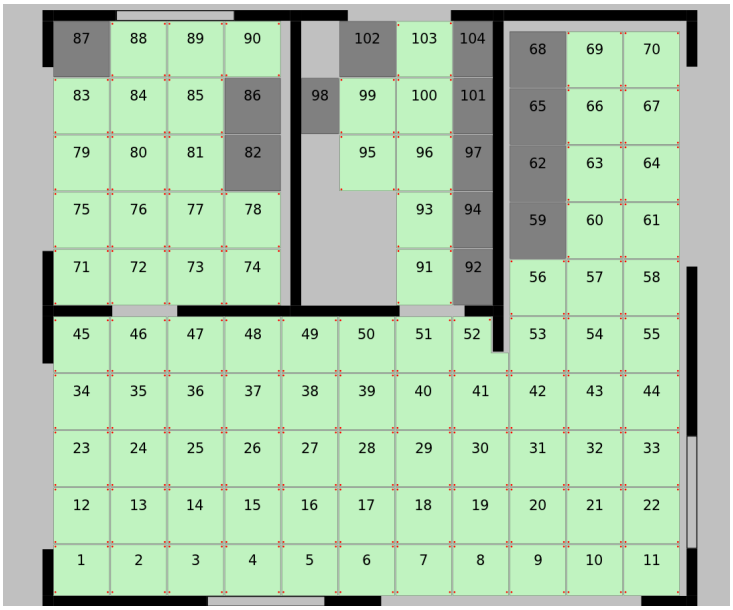


Figure 4.2 – A schematic view of the tiles composing the sensing floor, with their identifiers. The red dots indicate the positions of the sensors. The walls are shown in black. The tiles under the furniture in the sleeping room (top left room), under the toilet in the bathroom (top middle room), and under the kitchen furniture (top right room) have been left unequipped with sensors, and are shown here in grey.

4.1.1 Processing the raw sensing-floor data

The data generated by the floor can be passed to the software either as a recorded .txt file, or through a ROS publisher/listener interface by publishing it on the corresponding topic. The data processing software treats the data packets sent by the tiles, which have the format indicated in Tables 3.1 and 3.2, on page 33. The data packets arrive in disorder, as the network provides no guarantees on the order of arrival of packets. In addition, although the tiles are synchronised using Network Time Protocol, there is still some desynchronisation between the tiles in the order of milliseconds.

Each tile sends data packets regularly, every 20 milliseconds (50 Hz). The received packets are grouped by their recording time, generating a frame of the entire floor every 20 ms (see Fig. 4.3). For real-time processing with a limited amount of memory, the oldest frames are removed to free space for the frames to come. This guarantees an upper bound of memory consumption.

The working principle for the modules performing the segmentation, tracking and localisation of objects on the floor will be covered in Chapter 6.

4.1.2 The floor-data player

The player allows to visualise the pressure data recorded by the floor, and aggregated by the data processing software (see Fig. 4.3). For displaying the tiles and their sensors, it uses prior knowledge about their positions. The tiles that have sent data in the time window that corresponds to the visualised floor frame are present in the graph. The ones that have failed to send data in this time window are absent. The timeline extends itself in accordance with the length of the recording.

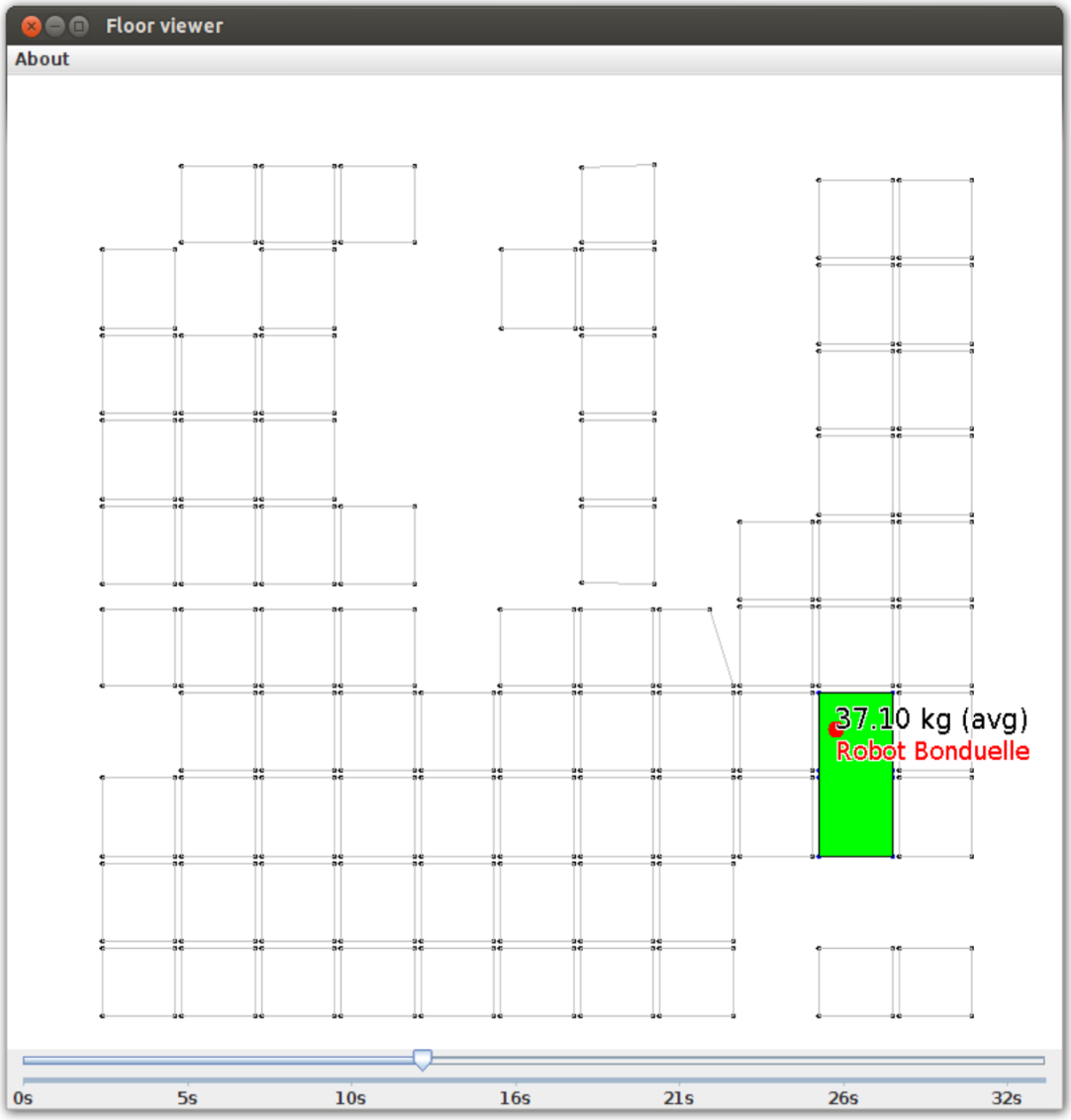


Figure 4.3 – Screenshot of the floor data processing software. The sensors are shown in black, interconnected here with their neighbouring sensors from the tile. The displayed positions of sensors and tiles correspond to their real location (see Fig. 4.2). Some tiles failed to send their data in the given 20 ms time frame, and are shown as missing here. A localised robot is displayed in green, together with its measured mass, and the location of its center of pressure (COP). A timeline on the bottom allows to scroll through the recorded floor frames.

4.2 Calibration of the tiles

Before performing any processing of the floor pressure data, a calibration of the pressure sensors has to be performed, that would determine the function converting arbitrary sensor units to newtons. The conversion function is located in a two-dimensional plane, where one axis corresponds to the ground truth load measurements in newtons, and the other one corresponds to the load measurements in arbitrary sensor units. As presented in Section 3.2.5 on the linearity of load measurements, the data conversion function for transforming the perceived conventional units of pressure into newtons is linear. This implies that the conversion function can be determined by identifying a point through which it passes, and its slope. Two calibration values are enough to determine the linear data conversion function on each tile: one value is obtained when recording the measured pressure when there is nothing on the floor, and the other one can be obtained by placing on each tile a known calibration weight. This calibration procedure is illustrated in Fig. 4.4.

Ideally, each sensor should be calibrated separately. This involves the precise positioning of the calibration weight on the tile, in a place where the distribution of the exerted load onto the supporting sensors is known in advance. For instance, a load placed in the center of a square tile is expected to be equally distributed among the sensors located in the corners of the tile. Considering the large number of tiles contained in a floor, and the time it takes to precisely position a weight, the procedure can become lengthy. A quicker joint calibration of the 4 sensors of a tile can be done, assuming that they all have the same slope of their functions converting arbitrary sensor units to newtons.

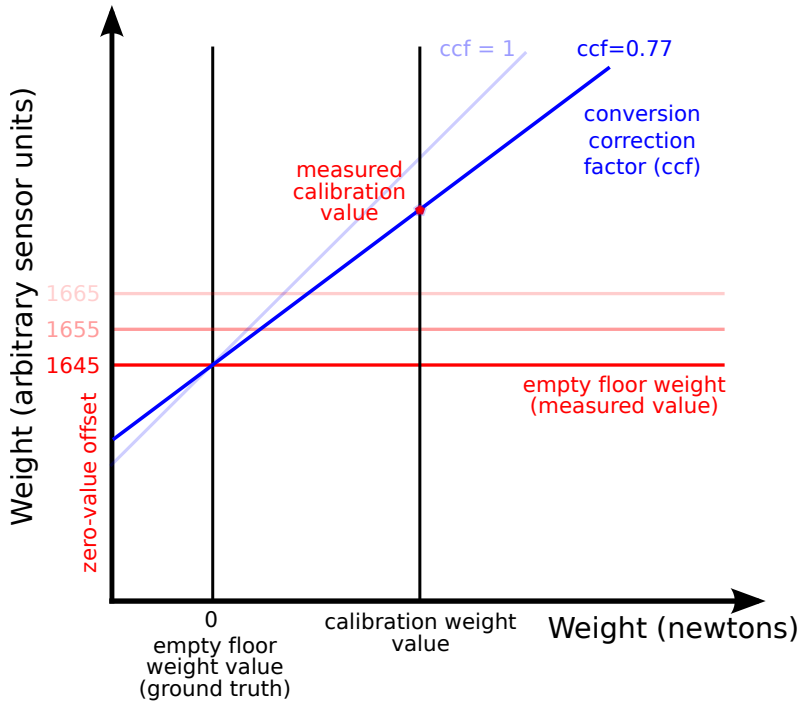


Figure 4.4 – Graphical illustration of the calibration procedure. First, the weight of the empty floor is measured, to identify the zero-weight offset. Then, the calibration weight is used to identify the slope of the function that converts arbitrary sensor units to newtons. In our case, the conversion correction factor is 0.77, which corresponds to 1 sensor measurement unit per 1.3 newtons of force.

The calibration weight can even be that of a static person, measured in advance on high-precision scales. Thus, if the sensors' zero-values that correspond to the state of an empty floor are recalculated at each launch of the algorithm, a quick calibration can be made by using the recorded data of a person walking and stopping on each tile. The recorded values are then used to calculate the constant correction factors applicable for the four sensors of each tile.

4.3 Calculation of the center of pressure on a tile

The processing of floor pressure data often involves the localisation of objects on the floor. Given the rigidity of the tiles forming the modular sensing floor, even if several pressure forces are applied on a tile, only the resultant force can be measured by the sensors underneath the tile. As mentioned earlier in Section 3.2.7, the floor can locate the position of a punctiform pressure exerted on a tile by calculating the center of pressures measured by the load sensors, using the formula:

$$\text{Center of Pressure} = \frac{1}{\sum_i^{\text{total sensors}} |\vec{R}_i|} \sum_i^{\text{total sensors}} \left(\vec{R}_i \times coords(Sensor_i) \right) \tag{4.1}$$

where $\vec{R}_i$ is the reaction force measured by the ith sensor supporting the tile, and the "$\times$" symbol is the *vector product* operator. Equation 4.1 is obtained from the static equilibrium equations, as further detailed in section 4.4.3.

This identification of a force's value and point of application on a tile when given the sensors' measurements, which is used in the processing of sensing data, has a complementary, inverse problem. The inverse problem, which is that of calculating the distribution of a pressure exerted on a tile onto the sensors supporting it, is relevant for simulation purposes, as we will se in the next section.

4.4 Inverse problem: calculating the load distribution on the sensors

For simulation purposes, it is relevant to compute the distribution of a pressure exerted on a tile on the sensors supporting it. This is useful, for instance, when combining human gait models with a simulation of the sensing floor, in order to obtain simulated floor pressure data. The problem can be formulated as follows: knowing the shape of the tile, the coordinates of the load sensors beneath the tile, and given a pressure force applied on the tile at a given location, how to calculate the values of the forces exerted on each sensor? The number of unknown reaction forces is equal to the number of load sensors underneath the tile. We therefore need as many independent equations as there are unknown forces, in order to identify their values.

Let us pursue this analysis of the distribution of forces on a support, by starting with the simplest form of a 2D support, a beam, and gradually increment its complexity.

4.4.1 Statically determinate beam

A beam is said to be in a statically determinate equilibrium when all the reaction forces on the beam can be calculated using the equations of *mechanical equilibrium*. These equations are:

- the vector sum of all forces is zero (applied forces and reaction forces);

- the sum of all torques is zero.

In a 2D setting, in the case of a force applied perpendicularly to a beam, the number of available equations equals 2, which imposes a limitation on the number of unknown variables (values of reaction forces) that can be identified. An example of a statically determinate beam is given in Fig. 4.5. The available equations are:

$$\begin{pmatrix} 1 & 1 \\ AB & -CB \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \begin{matrix} \text{Sum of forces equals zero} \\ \text{Sum of torques equals zero} \end{matrix} \tag{4.2}$$

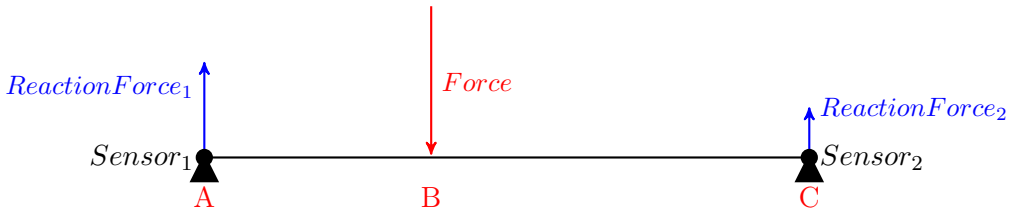


Figure 4.5 – Reaction forces on a statically determinate beam

In a frame of reference with the origin set in point B, the reaction forces can be calculated as follows:

$$\begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ AB & -CB \end{pmatrix}^{-1} \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{4.3}$$

$$\begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \frac{-1}{(AB+CB)} \begin{pmatrix} -CB & -1 \\ -AB & 1 \end{pmatrix} \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{4.4}$$

4.4.2 Statically indeterminate beam

A structure is considered *statically indeterminate* when the static equilibrium equations are not sufficient to find the reaction forces on the structure, because there are too many unknowns. A simple example is shown in fig. 4.6, where only 2 equations can be extracted (the sum of forces equals zero and the sum of torques equals zero). This is insufficient for calculating the 3 reaction forces arising from the supporting points of the beam. The problem is underconstrained and has an infinite number of solutions. Therefore, additional constraints have to be introduced in order to solve this system.

Several methods exist for solving this problem, such as: the flexibility method, the slope deflection method, the method of sections (for truss structures), and the moment distribution method [37] (for statically indeterminate beams and frames). For instance, the *slope deflection method* employs equilibrium equations for each node of the structure, in terms of deflections and rotations. It then uses moment-displacement relations to calculate the moments of force, and reduce the structure to a determinate structure. Let us now continue to the 3-dimensional case.

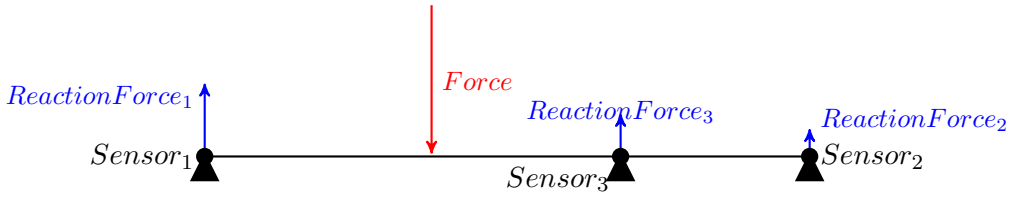


Figure 4.6 – Reaction forces on a statically indeterminate beam

4.4.3 Weight distribution on a triangular tile

In a 3-dimensional environment, we have 6 equations available for calculating the conditions of static equilibrium: the sum of forces equals zero (in 3 projections), and the sum of torques equals zero (in 3 projections). However, in the case of a force applied perpendicularly to the plane of a tile, the projection of forces on two of the axes (those in the plane of the tile, i.e. x and y) is equal to zero. In addition, as all the forces are vertical (they are parallel to the z-axis), there is no torque around the vertical, z-axis.

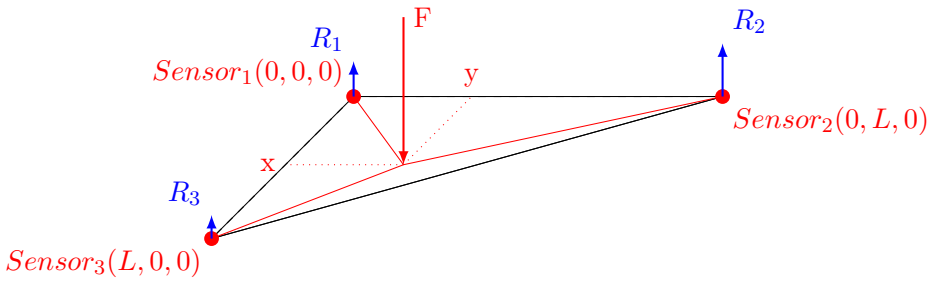


Figure 4.7 – Load distribution on an isostatic tile

Let us consider a given frame of reference, Ω. Let $R_i = [a_i, b_i, c_i]^T$ be the force vectors measured by the sensors $i = 1..N_c$ in the frame of reference Ω, with N_c being the number of sensors ($N_c = 3$ in the analysed case). These force vectors are respectively measured at points $\mathbf{P_i}$ with coordinates $P_i = [u_i, v_i, w_i]^T$ in the frame of reference Ω.

Let $F = [d_j, e_j, f_j]^T$ be the force vectors applied on the tile in the frame of reference Ω, $j = 1..N_f$, with N_f being the number of forces applied on the tile (or the number of points of support of an object placed on the tile) ($N_f = 1$ in the analysed case). These force vectors are respectively applied at points $\mathbf{U_j}$ with coordinates $U_j = [x_i, y_i, z_i]^T$ in the frame of reference Ω.

The static equilibrium of forces and torques can be formulated as:

$$\left(\sum_{i=1}^{N_c} R_i + \sum_{j=1}^{N_f} F_j\right) = \begin{pmatrix}0\\0\\0\end{pmatrix} \tag{4.5}$$

$$\left(\sum_{i=1}^{N_c} P_i \times R_i + \sum_{j=1}^{N_f} U_j \times F_j\right) = \begin{pmatrix}0\\0\\0\end{pmatrix} \tag{4.6}$$

where the "$\times$" symbol is the *vector product* operator. For $N_c = 3$, and $N_f = 1$ we obtain the following equations expressed at the point $(0, 0, 0)^T$:

$$\begin{pmatrix} \vec{R_1} + \vec{R_2} + \vec{R_3} + \vec{F} \\ \vec{P_1} \times \vec{R_1} + \vec{P_2} \times \vec{R_2} + \vec{P_3} \times \vec{R_3} + \vec{U} \times \vec{F} \end{pmatrix} =$$

$$\begin{pmatrix} a_1 + a_2 + a_3 + d \\ b_1 + b_2 + b_3 + e \\ c_1 + c_2 + c_3 + f \\ -b_1w_1 - b_2w_2 - b_3w_3 + c_1v_1 + c_2v_2 + c_3v_3 - ez + fy \\ a_1w_1 + a_2w_2 + a_3w_3 - c_1u_1 - c_2u_2 - c_3u_3 + dz - fx \\ -a_1v_1 - a_2v_2 - a_3v_3 + b_1u_1 + b_2u_2 + b_3u_3 - dy + ex \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.7}$$

In our case, the tiles are considered as rigid and flat. By placing the frame of reference Ω in such a way that its $\vec{z}$ axis is perpendicular to the plane of the tile, we obtain that $z = w_1 = w_2 = w_3 = 0$. In addition, the forces F and reaction forces R_i applied on the tile are perpendicular to its plane. Only the force components following the $\vec{z}$ axis of the force vectors are non nil. Thus, we will only consider equations 3, 4, 5 of the system of equations 4.7. This leaves us with the following system of equations:

$$\begin{pmatrix} c_1 + c_2 + c_3 + f \\ c_1v_1 + c_2v_2 + c_3v_3 + fy \\ -c_1u_1 - c_2u_2 - c_3u_3 - fx \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.8}$$

Let us rewrite again this equation, by reminding the meaning of each variable:

$$\begin{pmatrix} |R_1| & & + & |R_2| & & + & |R_3| & \\ |R_1| & \cdot S_{1y} & + & |R_2| & \cdot S_{2y} & + & |R_3| & \cdot S_{3y} \\ |R_1| & \cdot S_{1x} & + & |R_2| & \cdot S_{2x} & + & |R_3| & \cdot S_{3x} \end{pmatrix} = \begin{pmatrix} |F| \\ |F| \cdot y \\ |F| \cdot x \end{pmatrix} \tag{4.9}$$

where $|R_i|$ are the scalar values of reaction forces, $|F|$ is the scalar value of the applied pressure, and S_i are the sensors that support the tile. This can be refactored and rewritten as the following matrix equation:

$$\begin{pmatrix} 1 & 1 & 1 \\ S_{1y} & S_{2y} & S_{3y} \\ S_{1x} & S_{2x} & S_{3x} \end{pmatrix} \begin{pmatrix} |R_1| \\ |R_2| \\ |R_3| \end{pmatrix} = \begin{pmatrix} |F| \\ |F| \cdot y \\ |F| \cdot x \end{pmatrix} \tag{4.10}$$

This equation allows to calculate the pressure distribution on 3 supports (see Fig. 4.7). It is an equation having the form $A \times R = F$, where matrix A (containing the positions of the sensors) and matrix F (containing the intensity of the exerted pressure on the tile, and its position) are known. The matrix R, containing the values of the forces that act on the sensors, is unknown.

The value of the R matrix can be obtained by rewriting the equation in the following way: $R = A^{-1} \times F$. A solution to this equation exists only if the A matrix is invertible. The matrix A is invertible if and only if its determinant is not equal to 0. The determinant of matrix A shown in equation 4.10 is given by

$$1 \cdot \left(S_{2y} \cdot S_{3x} - S_{2x} \cdot S_{3y}\right) - 1 \cdot \left(S_{1y} \cdot S_{3x} - S_{1x} \cdot S_{3y}\right) + 1 \cdot \left(S_{1y} \cdot S_{2x} - S_{1x} \cdot S_{2y}\right) =$$

$$S_{2y} \cdot S_{3x} - S_{2x} \cdot S_{3y} - S_{1y} \cdot S_{3x} + S_{1x} \cdot S_{3y} + S_{1y} \cdot S_{2x} - S_{1x} \cdot S_{2y} \tag{4.11}$$

If we consider that the sensor S_1 is located in the origin of our frame of reference, namely $(0,0,0)^T$, we obtain:

$$Det(A) = S_{2y} \cdot S_{3x} - S_{2x} \cdot S_{3y} \tag{4.12}$$

If we further consider that the sensor S_2 is located on one of the axes of our frame of reference, having one of its coordinates equal to 0, we obtain:

$$\begin{aligned} Det(A) = &\quad S_{2y} \cdot S_{3x} && \text{if } S_{2x} = 0, \text{ or} \\ Det(A) = &\quad -S_{2x} \cdot S_{3y} && \text{if } S_{2y} = 0 \end{aligned}$$

This determinant equals zero only if the surface of the tile is zero. Given that the surface of our tile is non zero, the matrix A is always invertible. Therefore, the distribution of the forces applied to a point on the tile can be calculated for any given triangular tile, that is supported by sensors located in its corners.

Example for a particular triangular tile

In the case of the triangular tile presented in Fig. 4.7, the reaction forces can be calculated by applying the equations of static equilibrium:

- The sum of external forces applied to an object is equal to the null vector.

$$\sum \vec{F}_{ext} = \vec{0} \tag{4.13}$$

  In the case of a tile, this means:

$$\sum_i \vec{R}_i = -\vec{F} \tag{4.14}$$

- The sum of torques exerted at points P_i with respect to to a random point O is equal to the null vector. As a reminder, the moment of a force (or the torque) R_i, applied at a point P_i, calculated with respect to a point O, is given by

$$\overrightarrow{OP_i} \times \vec{R}_i \tag{4.15}$$

  where the "$\times$" symbol is the *vector product* operator. In the case of a triangular tile supported by 3 sensors, we have three reaction forces, R_1, R_2, R_3, applied at points P_1, P_2, P_3, and the force applied on the tile with the resultant force F, located in the center of pressure G. We therefore have:

$$\sum_i \left(\overrightarrow{OP_i} \times \vec{R}_i \right) = \overrightarrow{OG} \times \vec{F} \tag{4.16}$$

Otherwise put:

$$\begin{pmatrix} 0 \\ 0 \\ R_1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ R_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ R_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -F \end{pmatrix} \tag{4.17}$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ R_1 \end{pmatrix} + \begin{pmatrix} 0 \\ L \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ R_2 \end{pmatrix} + \begin{pmatrix} L \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ R_3 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ -F \end{pmatrix} \tag{4.18}$$

This gives us:

$$\begin{pmatrix}0\\0\\0\end{pmatrix} + \begin{pmatrix}LR_2\\0\\0\end{pmatrix} + \begin{pmatrix}0\\-LR_3\\0\end{pmatrix} = \begin{pmatrix}-yF\\xF\\0\end{pmatrix} \tag{4.19}$$

$$\begin{cases} R_1 + R_2 + R_3 = -F & (4.20)\\ LR_2 = -yF & (4.21)\\ LR_3 = -xF & (4.22) \end{cases}$$

We can therefore calculate the reaction forces:

$$\begin{cases} R_1 = F\left(-1 + \dfrac{x+y}{L}\right) & (4.23)\\ R_2 = \dfrac{-yF}{L} & (4.24)\\ R_3 = \dfrac{-xF}{L} & (4.25) \end{cases}$$

If there are more that 3 points of support (sensors in our case) under the tile, we have to solve an underdetermined linear system, that allows for an infinity of solutions. An approximate solution of Moore-Penrose type may be used in this case.

4.4.4 Weight distribution on a square tile

Each tile of the Inria SmartTiles prototype is supported by 4 load sensors, as shown in Fig. 4.8. Therefore, in order to identify the values of the forces that are measured by the load sensors, 4 equations are required. The previous section has shown that only 3 force equations are available in our case. This renders the tile *statically indeterminate* (or *hyperstatic*). Determining analytically the distribution of a force onto the sensors supporting the hyperstatic tile is not possible with the existing analytical tools, as the problem is underconstrained and has an infinite number of solutions. Techniques such as the *Finite Element Method* are usually employed to find approximate solutions to this problem, by considering the tile as being flexible.

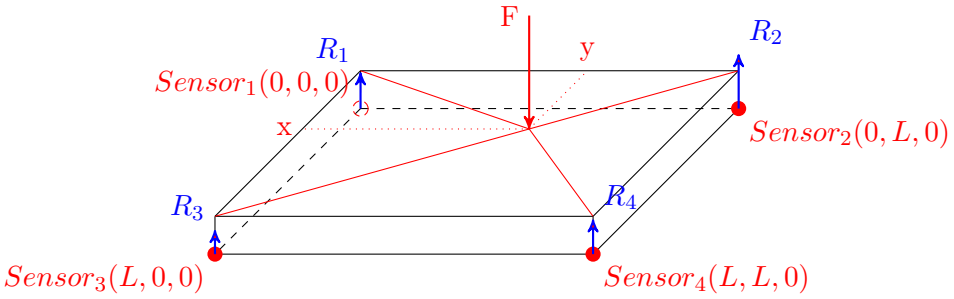


Figure 4.8 – Distribution of a punctiform pressure on a square tile with 4 supporting sensors.

4.5 Conclusion

This chapter has introduced the software developed for processing and simulating the pressure-sensing floor. We presented its architecture in terms of functional modules, that perform the parsing, sorting and storage of the incoming load data, and analyse it to detect, track and locate objects and humans. We also offered a graphical representation of the activities that take place on the floor, as perceived from the floor's perspective.

We introduced the calibration procedure for identifying the function converting arbitrary sensor units to newtons. The measured weight values can be interpreted as the mass of static objects located on the floor.

In our endeavour to create a simulator of the pressure-sensing floor, with the intent of coupling it to models of human activities, we have seen that it is not currently possible to analytically compute the distribution of a load on the sensors supporting a tile, if this tile is hyperstatic. However, approximate solutions can be obtained by considering the tile flexible, and applying methods such as the *Finite Element Analysis*. We conclude that in this respect, triangular isostatic tiles are more appropriate, but even they can have singularity points, which prevent the computation of an analytical solution to the problem of force distribution.

Part I has presented the existing prototypes of pressure-sensing floors, as well as the Inria SmartTiles prototype used throughout this thesis. We have seen the characteristics of the sensors used in our prototype, and the impact they have on pressure measurements and localisation of static and mobile objects. We introduced the data processing pipeline used to analyse the floor's raw data. It is now time to pursue onto an exploration of the capabilities of sensing floors.

Part II will present the services that a modular sensing floor can provide. We will first see how to extract the surface of an object in contact with the ground, with the goal of using it for object recognition (Chapter 5). Then, we will solve the problem of tracking and localising objects, robots and humans on our noisy, low-resolution sensing floor in Chapter 6.

Part II

Services provided by sensing floors

5

Object surface pressure scanning

Contents

5.1 Introduction

Ambient intelligence explores how sensing environments can interact with their inhabitants. We are interested in employing the sensors embedded in the environment for analyzing the activity of humans inhabiting it. In this sense, the recognition of objects with which humans interact plays an important role for the recognition of human activities.

In this chapter, we explore how to extract an object's surface in contact with the ground, with the goal of using it for object recognition. We present a pressure scanning technique, which employs sub-pixel shifting to make a series of low-resolution scans, which are then assembled into a high-resolution scan. It exploits the information contained in the differences between the shifted scans, and aggregates this information to compute a composite pressure scan.

The proposed pressure scanner architecture is composed of 4 load-sensing tiles. It calculates the weight transfer between the tiles when the analysed object slides over them. By using only the recorded mass and changements in the position of the center of mass, the scanner is able to reconstruct the contact surface of the object that slid on it. It also calculates the distribution of weight inside the surface of contact.

This chapter is organized into 4 sections. Section 5.2 presents the previous work on sub-pixel shifting and other related techniques. Section 5.3 describes the scanning equipment required for obtaining pressure scans, and introduces the algorithm that aggregates the scanning data. In

section 5.4, we describe our results obtained through simulation on a noiseless platform, as well as results obtained experimentally, on a physical platform subject to noise. Section 5.5 draws a conclusion and presents the perspectives of this work for home automation and its applications for industrial settings. This chapter is based on a publication [8] presented at the ICRA 2015.

5.2 Related work

High-resolution pressure scanning is usually performed using high-resolution pressure sensing devices (such as [96] and [148]). These often have a non-negligible price, due to the high number of embedded sensors. In this chapter we demonstrate that high-resolution pressure scanning is also feasible using a low-resolution sensor and techniques such as sub-pixel shifting, something which has not yet been attempted, to our knowledge.

Similarities can be identified between the pressure-sensing and imaging domains. The pressure perceived by a load-sensing surface is analogous to the amount of light perceived by an image-recording sensor. Both in imaging and pressure sensing, the sensor can be shifted by less than a pixel width to register a slightly different part of the incoming light or pressure. Two such sub-pixel shifted images, aligned along one of the two axes of the scan, can be assembled into a higher-resolution image, using the information gathered by the sub-pixel shift. Such imaging techniques that construct a high-resolution image from several low-resolution images containing sub-pixel shifts have been proposed by Peleg *et al.* [106], Keren *et al.* [69], Tekalp *et al.* [158] and Tom *et al.* [166].

The pressure sensor can also be composed of only one load-sensing tile, within a grid of other tiles that simply serve as physical support for the scanned object. Depending on how the object is placed on the sensing tile, the scanner will perceive new pressure data with every object placement that differs from the previous ones. Coupled with a system for identifying the displacement of an object's bounding box (e.g., a cartesian coordinate robot), it can be considered as the canonical scanner of such type, as it consists of a single detection element. A scanner equivalent in functionality while lacking the cartesian robot will be described in section 5.3.1.

The 1-sensing-tile scanner can be viewed as a single detection element scanner, for which the tiles that simply support the scanned object have the role of an aperture, that decides what portion of the object will be detected. This idea has been exploited in imaging, where a grid-like aperture was used to selectively allow light to pass through the aperture grid.

For instance, Huang *et al.* [61] used an LCD screen as an aperture in their implementation of the single detection element camera. The pixels of the LCD screen could turn transparent or non-transparent (black) to the incoming light, allowing different parts of the image to be perceived. The resolution of the obtained composite image was dependent on the resolution of the LCD screen used as apperture.

Both technologies (single detection element with a high-resolution aperture, sub-pixel shifts of the aperture) can potentially be combined to compute images with even higher resolution.

5.3 Methodology for high-resolution pressure sensing

5.3.1 Scanning equipment

The instrument we use for scanning is constructed of 4 isostatic load-sensing tiles (see Fig. 5.1). The benefit of isostatic load-sensing tiles is the capability to calculate unambiguously the

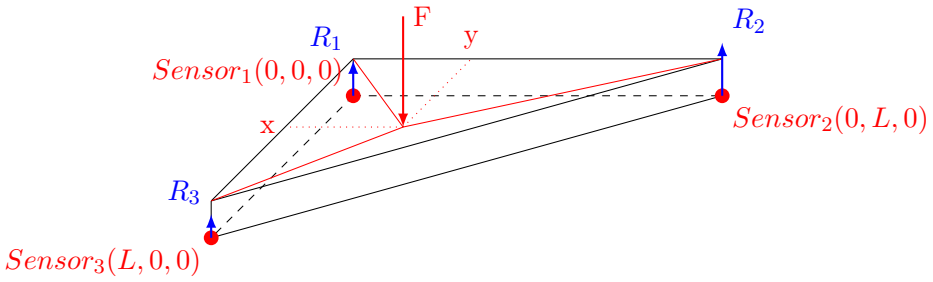
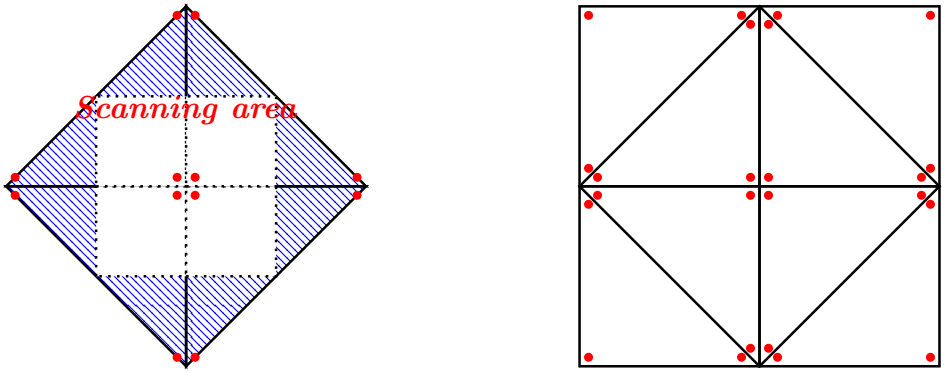


Figure 5.1 – Load distribution on an isostatic tile



(a) Scanner with 4 isostatic tiles, which employs only half of the tiles' scanning surface.

(b) Scanner with 8 isostatic tiles, which employs all the scanning surface of the tiles.

Figure 5.2 – Examples of scanners with isostatic load-sensing tiles. The load sensors are represented as red dots. The scanner in Fig. 5.2a requires fewer sensors per sensing pixel (3 instead of 6), as compared to the scanner presented in Fig. 5.2b. Using fewer sensors has the benefit of generating less uncertainty in the measured value.

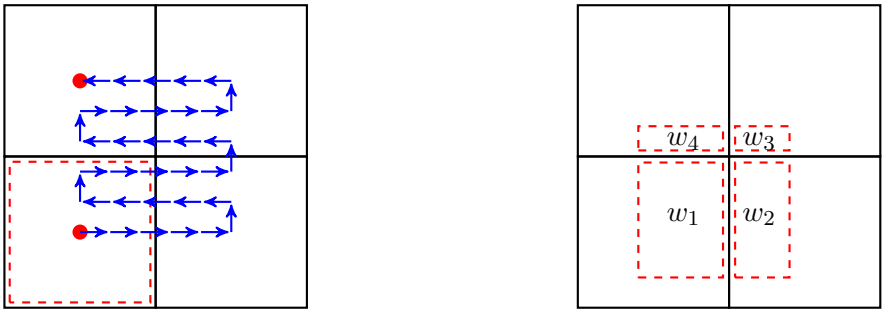
distribution of load on the sensors supporting the tile. The tiles are assembled so as to have vertical and horizontal frontiers between them, that will measure the flow of mass between the tiles (see Fig. 5.2). This type of scanner architecture can scan the contact surface of objects with size up to one fourth of the total scanning surface.

For clarity reasons, in the rest of the chapter we will consider that the scanner is composed of only 4 square scanning parts, corresponding to the effective scanning area seen in Fig. 5.2a.

5.3.2 Scanning procedure and its particularities

The object to be scanned is slid in a boustrophedon manner, i.e. with alternative horizontal and vertical translation movements over the plates (see Fig. 5.3a), allowing the entire object surface to be scanned. After each translation movement, the tiles measure the weight and calculate the coordinates of the center of pressure for the object portion standing on them (see Fig. 5.3b). Thus, at each step, the 4 tiles generate a low-resolution 4-sliced image of the load on the contact surface. These images are then assembled into a single grid-like high-resolution image of the contact surface (see Fig. 5.4b).

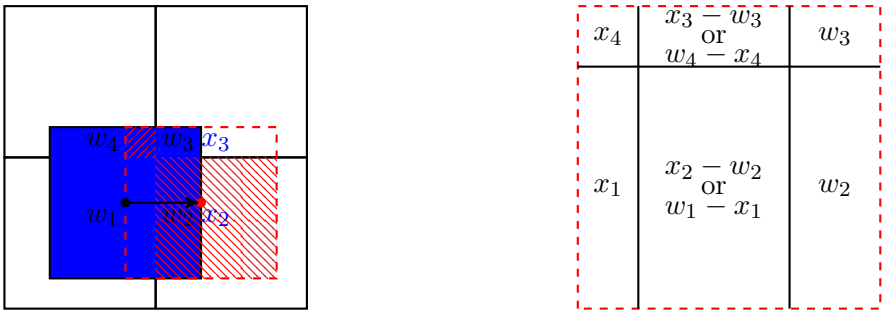
The displacement brought by each translation can be measured by tracking the center of pressure of the object on the scanning surface. The coordinates of the object's center of pressure



(a) Sliding the object in a boustrophedon manner over the scanning surface. The object's bounding box is highlighted in red.

(b) Each scan generates a 4-sliced image of the weight-distribution inside the object's bounding box.

Figure 5.3 – Object scanning methodology



(a) Overlapping of two low-resolution load images.

(b) Aggregate of two overlapping low-resolution load images.

Figure 5.4 – Construction of a high-resolution image from low-resolution images.

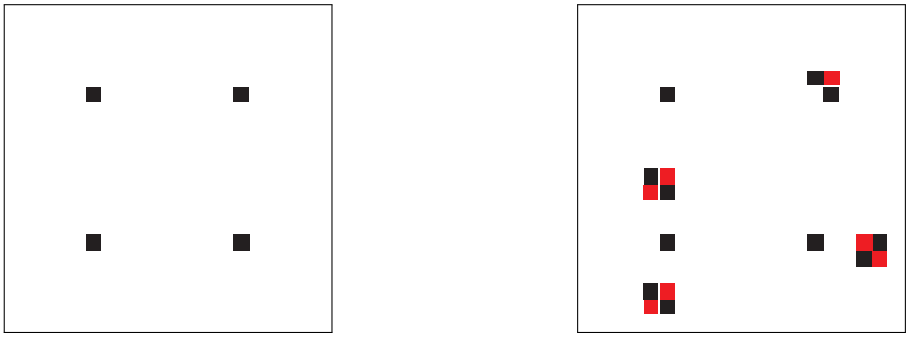
will correspond to the coordinates of the center of mass of a system of particles composed by the sensors of the scanner, where each particle's mass is proportional to the weight perceived by the corresponding sensor. The distance travelled by the object's center of pressure corresponds to the distance travelled by its bounding box.

The translation movements can be performed by a cartesian coordinate robot (like the one presented in [145]), if the goal is to achieve high precision. For less precise scanning, the object can even be translated by hand, and scanned once the translation movement is completed and no exterior forces act on the object.

The problem can be schematically reduced to determining the weight of each pixel in the load-image representing the contact surface of the analysed object (see Fig. 5.4). Any given two load images can be assembled into a higher resolution image if both are subdivided by a common x or y line. This line is used for aligning the two images, creating the additional high-resolution image subdivisions.

The scanning resolution can be modified by changing the object shifting step: bigger steps lead to less gathered information, and therefore to lower-resolution final scans, and vice versa. The maximum spatial resolution depends on three factors: the precision of the object translation, the sensitivity of the pressure sensor, and the sensor noise. The noise level of the sensor must be inferior to the recorded pressure value in each pixel. This means that a noisy sensor will be able to make high-resolution scans only for heavy objects.

The object translation movements must not be accompanied by rotations, as they skew



(a) The expected scan figure, without skewing.

(b) Example of scan image skewing, obtained while scanning with a 10% chance of rotating the figure by 5 degrees.

Figure 5.5 – Example of skewing a scan image by rotation of the scanned object. Positive pressure values are represented in grayscale, while negative pressure values are shown in shades of red. Skewing generates negative pressure values, that counterbalance the additional pressure points detected.

the scanned load images, preventing them from aggregating into a sharp, non-ambiguous load-distribution image. Considering the gradual calculation of the pressure scan image, the errors are cumulative, propagating themselves through the scan. Fig. 5.5 presents an example of skewing due to object rotation.

The scanning speed depends on the speed of object manipulation, and on the measuring frequency of the sensor. Due to the presence of sensor noise, several measurement must be made from which the most probable pressure value is calculated.

For example, a system able to translate the scanned object at a speed of 1 translation per second, and taking 1 second to measure the weight of the object on the scanner, scanning an object of size 1 m^2, with a resolution of 1 cm^2, would require 10 000 measurements, which would take 5.5 hours to record. Scanning the same object at a resolution of 10 cm^2 would require 100 measurements, and take only 3 minutes 20 seconds of time.

5.3.3 Scanning algorithm

Each scan of the object generates 4 pieces of information, provided by the 4 tiles of the scanner (see Fig. 5.3b), that contain: the coordinates of the scanned object portion, the weight of this object portion, and the center of pressure of this portion.

The scanned pieces of the object are stored in a list of type objectPiece. Each objectPiece contains 6 fields: the x and y axes that bound it (xLeft, xRight, yTop, yBottom), its weight, and the coordinates of the center of pressure. These object pieces are then assembled into a high-resolution image using the pressure-image registration algorithm described hereafter.

Each time a new piece of the object bounding box is scanned, the algorithm checks if this piece contains or is contained by another known piece. If the new piece contains another known piece, the surface and weight of the latter is subtracted from the new piece, thus reshaping the new object piece that has to be added to the scan. If the new piece is contained by another known piece, the latter is split in 2: the new piece, and the piece obtained by subtracting the new piece from the one already known.

Performing this type of scanning while moving the object in a regular manner generates a pressure grid. Fig. 5.6 offers an intuition on the way the algorithm executes itself.

The pressure-image registration algorithm is presented in pseudocode in Algorithm 5.1. Each new scanned portion of the object's bounding box is checked whether it is contained by another portion already scanned, and whether they have common boundaries, so as to eventually subtract one from another and obtain the portion containing the additional information brought by the scan. Thus, the scanning algorithm's worst case complexity is quadratic in the number of object pieces already scanned.

Algorithm 5.1 – Pressure-image registration algorithm

```
addPiece(newObjPiece)
{
   if (newObjPiece.hasZeroSurface)
      return;
   else if (listOfObjPieces.isEmpty)
      listOfObjPieces.add(newObjPiece);
      return;
   // if "listOfPieces" is not empty
   else
      // loop through all pieces in
          listOfPieces
      for each (objPiece in
          listOfObjPieces)
         if (newObjPiece.contains(objPiece))
            croppedObjPiece =
                newObjPiece.subtract(objPiece);
            addPiece(croppedObjPiece);
            return;
         else if
             (objPiece.contains(newObjPiece))
            croppedObjPiece =
                objPiece.subtract(newObjPiece);
            // Split and remove the piece
                containing the pieceToAdd
            listOfObjPieces.remove(objPiece);
            // Add the splits
            listOfObjPieces.add(newObjPiece);
            listOfObjPieces.add(croppedObjPiece);
            return;
         end if
      end for
   end if
}
```

Algorithm 5.2 – Boustrophedon pressure scan

```
Boustrophedon_scan()
{
// 1. Traverse
// 1.1 Find movement direction
// 1.2 Move in that direction until the end
// 2. If can go down, move down and repeat}
   repeat
      scanObject();
      if (!canMoveObjectLeft())
         while (canMoveObjectRight())
            moveObjectRight();
            scanObject();
            addPiece(topLeftPiece);
            addPiece(topRightPiece);
            addPiece(bottomLeftPiece);
            addPiece(bottomRightPiece);
         end while
      else
         while (canMoveObjectLeft())
            moveObjectLeft();
            scanObject();
            addPiece(topLeftPiece);
            addPiece(topRightPiece);
            addPiece(bottomLeftPiece);
            addPiece(bottomRightPiece);
         end while
      end if

      if (canMoveObjectDown())
         moveObjectDown();
      end if
   until (!canMoveObjectDown())
}
```

The boustrophedon algorithm simply moves the object as shown in Fig. 5.3a, and generates the corresponding scan after each translation. This procedure, that allows all the sections of the object to be scanned, is detailed in Algorithm 5.2. The boustrophedon algorithm has a complexity of $\mathcal{O}(1)$ in the desired scan resolution.

5.4 Experimental results

Experiments have been done both using a simulator of load-sensing tiles that we developped for our work on sensing floors for ambient intelligence, and using a real-world implementation of the scanner. These experiments are detailed in the following sections.

5.4.1 Scanning simulation

For simulation purposes, we developed a software tool for quick prototyping, written in Java, in which the simulated scanner was represented by 4 load-sensing tiles, as presented in section 5.3.1. The object to be scanned was represented using a set of pressure points, which correspond to the surface of the object in contact with the scanner (see Fig. 5.7a). Each point had its own pressure intensity value, and its coordinates inside the object's bounding box. The software calculates the total amount of pressure exerted on each tile of the sensor.

By translating the object following a boustrophedon trajectory, we obtain a pressure scan of the object's surface in contact with the scanner, as shown in Fig. 5.7. Modifying the object translation step influences the resolution of the final scan, as can be seen in Fig. 5.7b. The scan result for an object with different pressure intensities inside its contact surface is presented in Fig. 5.8. These pressure differences were represented as grayscale intensities in the final result.

Given that the sensing tiles can not only weigh the portion of the object they support, but also calculate the center of pressure for that portion, the pressure scans can be augmented using information about the centers of pressure in each cell of the pressure-grid scan. There are no limitations on the shape of objects that can be scanned. The scan of a sample object with non-convex shape of its contact surface is shown in Fig. 5.8.

5.4.2 Physical experiment

The physical experiment was implemented on a scanner composed of the square sensing tiles presented in chapter 3. Although these tiles are non-isostatic, this has not hindered the experimental results.

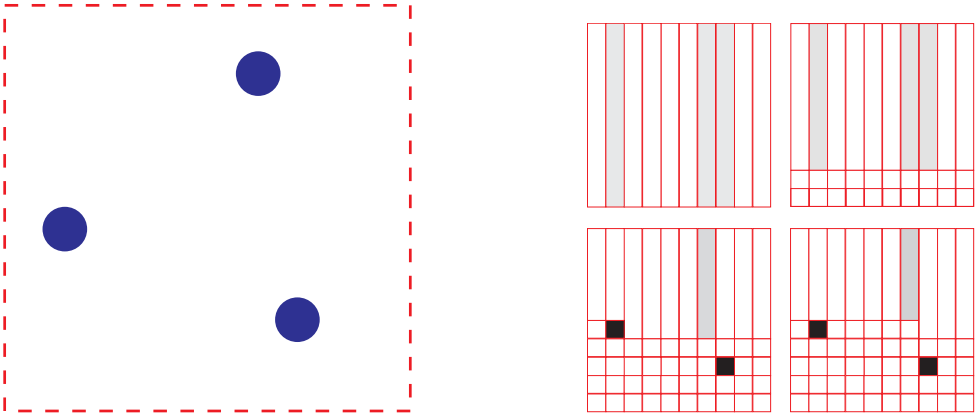
The scanned object was a chair, weighing 5 kg, and with the chair legs forming a trapezoid shape (see Fig. 5.9a). Given the high level of noise on our platform ($\pm$2.5 kg), the chair was loaded with additional 40 kg of weight disks, to generate a pressure scan that would be easily discernable from noise.

First, the tiles were calibrated using a standard weight, to ensure that they all converted the measured forces into pressure units in the same way. After this, the average zero-pressure value was recorded for each load-sensing tile when no objects were placed on the scanner. This served as reference for the ulterior measurements.

The scanned object was then placed on the scanner and weighed by each load-sensing tile. The pressure values for each of the 4 tiles were calculated by averaging the pressure values over 50 measurements (which takes 1 second at 50 Hz with the current prototype). The position of the object was measured by hand using the millimeter paper. It could not be done automatically by calculating the position of the center of pressure, because the load sensors were subject to noise, which would have prevented the alignment of measurements into a grid. The translations were carefully done by hand, which also constituted a possible source of error.

Due to the manual translation required for each measurement, the experiment, which consisted of 100 measurements, lasted for 2 hours. The duration of the experiment, combined with the load placed on the tiles (45 kg) introduced hysteresis-related errors towards the end of the experiment. However, according to the sensor specification, this error is bounded to 0.03% of the full scale measurement, and is thus negligible.

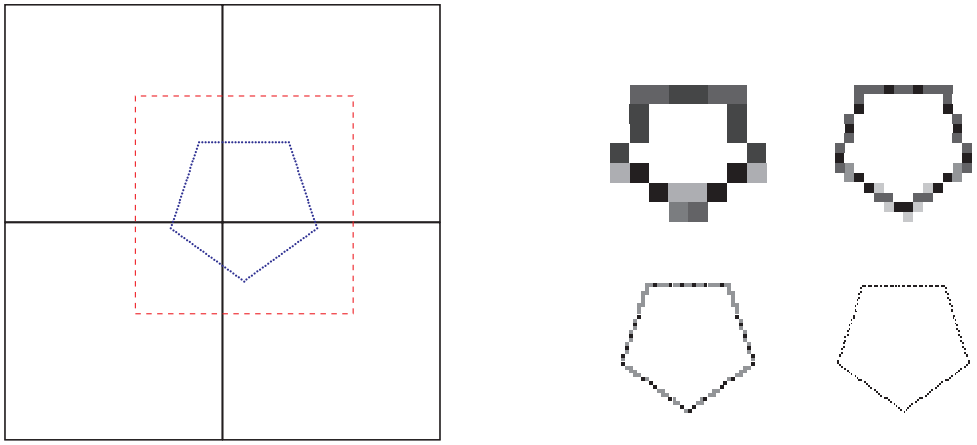
The results of the scan are presented in Fig. 5.9c and 5.9d. The noise present in the scan is related to the sensor noise, and was removed using a rough threshold filter, with a threshold set at 2x the noise of an individual sensor. Noise filtering done using a statistical model of the sensor noise should yield even finer results.



(a) The surface to be scanned.

(b) Scanning phases. The ratio of weight to surface size is color coded in grayscale.

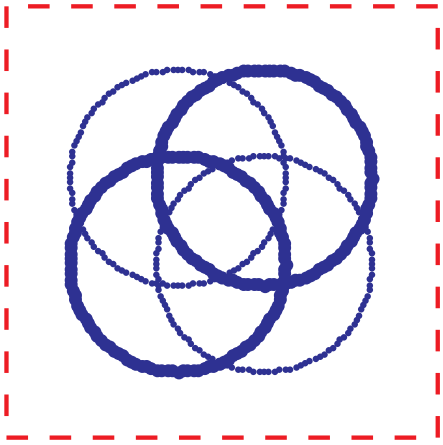
Figure 5.6 – Sample scan execution on a simulator. As the scanning progresses, the pressure image gets richer in detail.



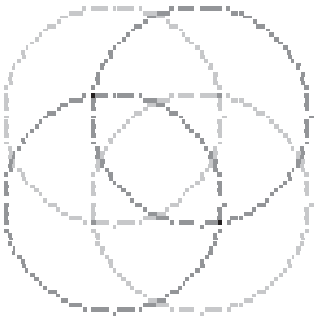
(a) An object's points of support on the pressure scanner. The object's bounding box is shown in red.

(b) Scanning results for the object presented on the left, at resolutions of 0.1, 0.05, 0.02 and 0.01 of object size.

Figure 5.7 – Scan simulation performed at different scanning resolutions.



(a) An example with four crossed circles, two of which are 2 times heavier than the other ones. Object thickness represents weight here, and is otherwise not representative.



(b) The resulting pressure scan reflects the pressure differences between the circles, in shades of gray.

Figure 5.8 – Scan simulation for an object with non-uniform pressure distribution inside its surface in contact with the floor.

Unfortunately, the sensor noise on our current platform (±2.5 kg) did not allow us to scan objects with a more complex surface (containing more than 10 pixels), as this would have required them to be too heavy to manipulate.

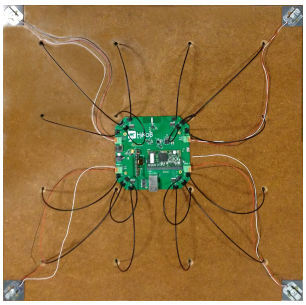
5.4.3 Open questions

There is no proof that the boustrophedon trajectory is optimal in terms of highest gain of information in the lowest possible number of translations of the object. Finding the optimal (quickest) trajectory for scanning an object with the described sensor in order to find the geometry and the pressure in each point of its contact surface remains an open question.

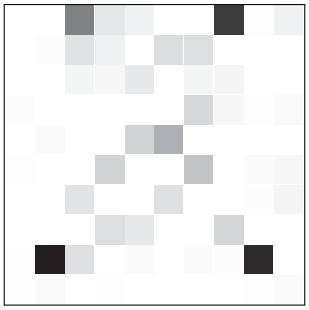
It would also be interesting to scan objects that don't move in a regular fashion, but which are augmented by an orientation sensor. These objects can be scanned at the moments when their orientation matches with the one required by the scanner.

Fine-grained noise filtering using a statistical model of the noise is also of great interest for these platforms. It would allow to greatly improve the accuracy of measurements, enabling pressure scanning for more lightweight objects.

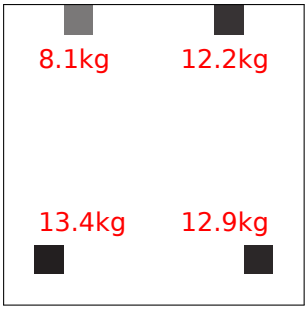
(a) A chair being scanned on a scanner made of 4 load-sensing tiles. One tile was covered in millimeter paper, to be used as a location reference. Translations were performed by hand.



(b) The underneath of the load-sensing tiles used, showing the load sensors located in the corners.



(c) The pressure scan obtained without noise filtering. Most of the noise is caused by sensor noise. The highest pressure is encoded as black.



(d) The pressure scan obtained with noise filtering. The perceived pressure is rendered as an equivalent weight value at 1G acceleration.

Figure 5.9 – Physical implementation of the pressure scanner.

5.5 Conclusion and perspectives

This chapter presented a new type of high-resolution pressure scanning technique, which employs sub-pixel shifting on a low-resolution load sensor to generate and assemble a high-resolution pressure scan. The proposed scanner architecture is composed of 4 load-sensing tiles, on which the scanned object is translated vertically and horizontally in a boustrophedon manner using sliding moves. A proof-of-concept was provided, together with a set of experimental results obtained on a noiseless simulator and on a physical implementation of the scanner.

This system could be used for measuring the weight distribution on surfaces. For instance, on transportation lanes, it can check for abnormalities in weight distribution between the front and rear driving axles of vehicles. The same applies for railroad tracks, where it can measure the weight distribution of the rolling stock. In an industrial setting, this technique could be used for measuring the density in each slice of a block of metal or other material.

It can also be used as a floor sensor in sensing environments, recognizing mobile objects by their surface in contact with the ground. In the context of sensing environments equipped with floors consisting of this type of load-sensing tiles, this scanning system could provide a way for obtaining footprints, thus offering input data for human recognition algorithms.

Pressure surface scanning provides a new way of interpreting load data, which is distinct from the common *ground reaction force* profile interpretation. It improves weight-based object recognition, adding information about the floor projection of the object's shape.

In this chapter we have seen how a network of load sensors can be exploited to perform low-level data analysis for object recognition. The next chapter will present how to process load-sensing data on a higher level, in order to detect, track and recognise multiple entities (objects and humans) evolving on a sensing floor.

6

Detecting, tracking and identifying humans, robots and objects

Contents

The reconstruction of a model of activities inside an environment requires the localisation, tracking and recognition of static and mobile entities inside it. A non-intrusive way to do this is by employing load-sensing floors [3, 110, 138]. The traditional way of recognizing humans was by first tracking them, extracting gait features and then identify them using clustering techniques [104] or HMM [3, 110]. However, this type of recognition failed whenever the extraction of gait features became impossible. This happens when multiple users walk alongside, preventing the algorithm from correctly segmenting and tracking each of them on the floor.

This chapter introduces an object detection, tracking and recognition technique, which localises and recognises multiple objects simultaneously by analysing the load they exert on the floor. As it does not extract gait features for recognition, it does not require fine tracking of individual persons inside a group. Inspired by computer vision, this technique processes the floor pressure-image by segmenting the blobs containing objects, tracking them, and recognising their contents through a mix of inference and combinatorial search, using information about object weight and size. It can be used to provide a probabilistic input for multi-modal object tracking and recognition systems. The concept was successfully validated in the context of daily life activities, inside an ambient intelligence setting, where the non-intrusive load-sensing floor presented in chapter 3 was used. The experimental scenarii involved multi-object tracking and recognition on low resolution sensors, crossing of user trajectories, and weight ambiguity between

objects. The main drawback of this approach is its computational complexity, due to the sheer number of possibilities of correlating known objects to the observations made. However, this issue is classically solved using dynamic programming, which is efficient in this case because the most probable object tracking and localisation solutions can be quickly identified and reused for further generation and evaluation of object localisation hypotheses.

The rest of this chapter is organised as follows. Section 6.1 presents the state of art in the domain of tracking and recognition with load-sensing floors. In section 6.2, our load-data processing approach is exposed, with an emphasis on object detection, tracking, and recognition. Then, experimental results for the proposed algorithm are presented and analysed in section 6.3. Finally, directions for future work are evoked in section 6.4. This chapter is based on a paper [9] published in the *IEEE Sensors* journal.

6.1 Localisation using sensing floors: state-of-the-art

The sensing floors presented in chapter 2 were mainly designed for the localisation and recognition of humans and objects. The localisation capability was natively implemented into the floors through the spatialised aspect of the sensors composing the floor. Whereas object recognition was explored on high-resolution pressure sensors using the shape of objects' surfaces (e.g., to distinguish between walking barefoot and in shoes [96]), human recognition centered around the extraction and use of gait features.

Addlesee *et al.* [3] recognise humans using their footsteps' pressure-profile as data, and using Hidden Markov Models as classifiers. They also mention the problem of interpretation of spread loads, which prevents the segmentation of objects that span several tiles on a modular floor. Orr *et al.* [104] learn models of humans using the vertical ground reaction force profile, as well as its derivates: the maximal load value during heel strike and during toe push-off, and the minimal load value recorded during the weight transfer from heel to toe. Recognition is done using a *Nearest-Neighbor* algorithm in a multi-dimensional space.

Similarly, Pirttikangas *et al.* [110] recognised individual persons walking on the floor by using the pressure pattern of their gait and HMMs. Middleton *et al.* [91] employed a binary pressure-sensing floor, and used features like the stride length, stride cadence, and time-on-toe to time-on-heel ratio to recognise the subjects using a standard distance metric of similarity. Qian *et al.* recognised people on a high-resolution sensing floor (1 sensor per cm^2) using features extracted from their gait in [115].

Yun *et al.* employed multilayer perceptron networks to identify individuals based on the extracted gait features (stride length, foot angle, heel strike time, etc.) using their UbiFloorI and UbiFloorII binary pressure sensors [192, 194, 195, 193].

Vera-Rodriguez *et al.* employed both spatial features (footprint) [173] and the temporal features of footsteps (ground reaction force) [170] to recognise people on a high-resolution pressure-sensing floor, which employs piezoelectric sensors mounted on a printed circuit board [171].

Schmidt *et al.* [138] performed object recognition by querying a database of known objects whenever a change was detected in the total weight of a scene, to see if there exists an entry that has the same weight as the absolute difference in weight detected. However, this process was neither probabilistic in the processing of candidate solutions, nor could it detect simultaneous introductions or removals of objects from the scene.

Concerning object tracking on sensing floors, inspiration can be sought in the field of computer vision, where techniques such as Bayesian Filtering [48], Joint Particle Filtering [20], Probabilistic

Multi-Hypothesis Tracking [183], and Joint Probabilistic Data Association Filtering [47] have been applied for tracking multiple targets. Challa *et al.* provided an overview of these techniques in *Fundamentals of object tracking* [30].

Murakita *et al.* [98] performed multi-user human tracking on the VS-SS-F InfoFloor system using the Markov Chain Monte Carlo method. However, the employed floor sensors gave only a binary information about the occupation of its constituent tiles. Tracking would fail whenever two or more targets crossed their paths, generating tracking ambiguity. Attempts were made to solve this problem by fusing the information from the floor sensors with that from on-body acceleration sensors [63].

Savio and Ludwig identified footsteps on their *smart carpet* with binary capacitive sensors using clustering algorithms based on Maximum Likelihood Estimation and Rank Regression analysis, which allowed the extraction of user's trajectory [137].

Similarly, Suutala *et al.* [151] used Gaussian Process Joint Particle Filtering to track humans on a tiled floor equipped with binary switch sensors. Their algorithm did not use weight information for object tracking, as this information was not provided by their hardware. However, in the case of pressure-sensing floors, we can also exploit the weight information to evaluate the generated tracking hypotheses, a technique that we propose and exploit in this chapter.

The following sections will present an algorithm that detects, tracks and recognizes objects by using only the information about their size and weight. It offers a solution to the problem of interpretation of spread loads, when objects span several tiles on a modular floor. In comparison to the aforementioned tracking techniques, which exploit only binary data about the presence or absence of objects, our tracking algorithm exploits the weight data provided by load-sensing floors, as detailed in Section 6.2.3. As it tracks and recognises objects individually and in groups, it is more fault tolerant as opposed to algorithms that extract gait features, which require fine segmentation and tracking of targets. This technique can boost recognition when used complementarily with algorithms that extract features from the human gait, but can also serve as a gracefully degraded recognition mode whenever these fail. We have implemented our object recognition algorithm on the *SmartTiles platform* presented in chapter 3.

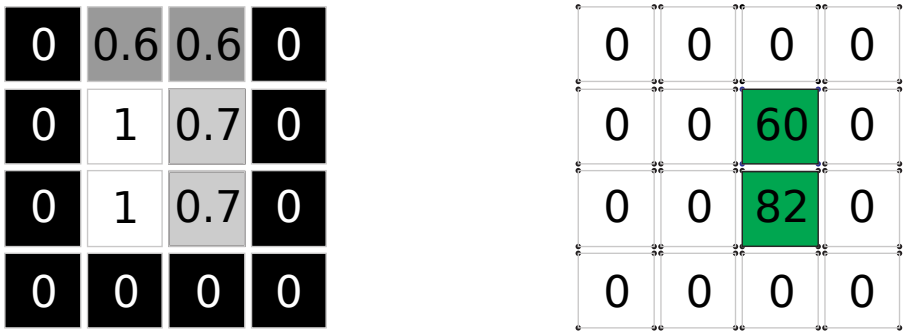
6.2 Methodology: load data processing flow

6.2.1 Similarities with video image processing

Parallels can be drawn between data processing in the context of computer vision and that of load-sensing floors. The field of view of a camera is analagous to the surface covered by a sensing floor. The light-intensity bitmap image generated by a camera subjected to light is analagous to the load image generated by a load-sensing floor under a static weight: the first encodes intensities of black (for monochrome images), while the second encodes pressure intensities (see Fig. 6.1). This hints that traditional image processing techniques can be employed to solve similar problems in the context of load-sensing floors.

The traditional data processing flow in computer vision usually consists of the following steps: background subtraction, blob detection, blob tracking, and blob recognition. The data processing flow that we propose for load-sensing floors is similar, and has the following structure: background subtraction, blob detection using *connected-component labeling*, and a feedback loop perfoming blob tracking and localisation of objects. The algorithm receives as input:

- the force values registered by the sensors composing the floor when there is nothing on it (i.e. the zero values used for background subtraction);



(a) The light intensity perceived by a photo sensor, bounded between 0 (black = minimum light level) and 1 (white = maximum light level).

(b) The pressure intensity perceived by the floor can be represented in scalar values, using only a lower bound (0 = no pressure).

Figure 6.1 – The similarity between floor pressure readings and light intensity recordings on a monochrome image.

- the values of forces recorded at time t, together with the coordinates of the load sensors that sensed them;
- a list containing the models of objects known to the floor, and which can be present on the floor.

The object models have the following structure: object name, mass (in kilograms), and length (in meters).

6.2.2 Background subtraction and object detection

Objects are detected on the floor by background subtraction and subsequent *connected-component labeling*. The background subtraction allows to process the data only from sensors that perceived force values above zero, filtering out all other sensors. Then, *connected-component labeling* [40] links together all sensors that are potentially supporting the same object, thereby forming blobs (see Fig. 6.2). It uses the length of the largest known object as a proximity threshold: if two sensors detected pressures over the noise threshold, and if the distance between the sensors is smaller than the size of the biggest known object, these are linked together, forming a connected component.

The size of the largest known object is calculated from the list of known object models. After this phase, any object present on the floor is guaranteed to be contained by one blob at most. On the other hand, a blob may contain one or several objects.

In our implementation, the sensor network is represented as a graph. The sensors which are left after background subtraction, are linked using *connected-component labeling*, forming blobs. Figure 6.2 shows the set of load sensors embedded into the floor, where each sensor is represented as a dot. The blobs detected by the floor were then overlayed onto this image.

For simplicity reasons, we will consider that there is no occlusion in our system, which occurs when a tile malfunctions and stops sending load data, or when a tile is not equipped with sensors.

6.2.3 Object tracking

After the detection of blobs on the floor, we can try to infer the objects located in these blobs by using their weight. However, the load force detected by the sensors oscillates during activities

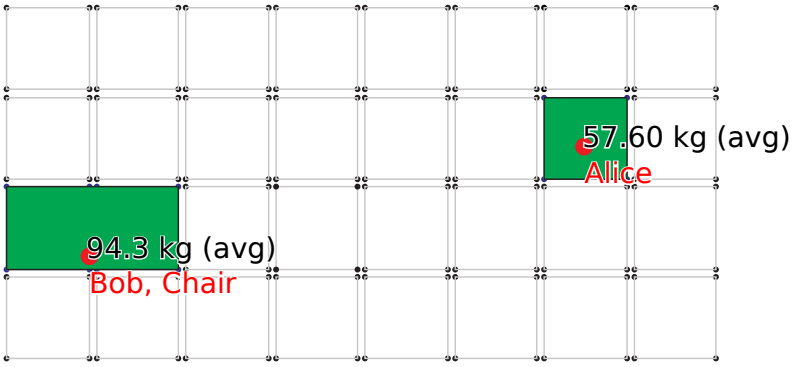


Figure 6.2 – Object recognition sample. The floor load sensors are represented as little black dots. The detected blobs are colored in green. The numbers in black show the average blob weight, calculated over a time window. The red dots show the position of blobs' centers of mass. The text in red shows the best recognition guess.

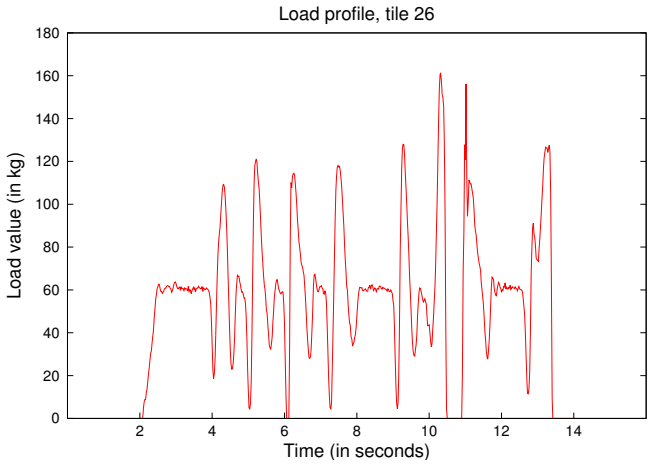


Figure 6.3 – The load profile of a person squatting and jumping on a load-sensing tile. Notice that the load oscillation is centered around the mass of the person, which is 60 kg.

such as walking or squatting and standing up, which involve body acceleration and deceleration that are transmitted to the floor (see Fig. 6.3 for an example). Thus, the value of this force cannot be directly converted into an estimation of an object's mass. Nevertheless, the value of this force oscillates around the weight of the object or person, as mentioned in [3]. Therefore, it is possible to approximate the total weight of objects inside a blob, by calculating the blob's average weight over a sliding window of time. This requires blob tracking.

An adequate solution to this problem is to use a tracking technique that takes into consideration the different ways in which blobs can evolve. A blob can appear in the scene, disappear, remain constant, merge with other blobs, split into several blobs, or it can exchange contents with another blob (see figures 6.4 and 6.5 for an example of merge and split events). We propose a method that explores the entire search space of joint blob evolution hypotheses (except for remote content exchange between blobs, rarely encountered in practice), and sorts these hypotheses according to a given criteria. Intuitively, the optimal solution should minimize the total distance travelled by the blobs inside the scene between two instants of time, as well as minimize the weight difference between the correlated blobs in two neighboring time frames. We define penalties for each type of blob evolution, which are used for ranking the tracking hypotheses (see Table 6.1).

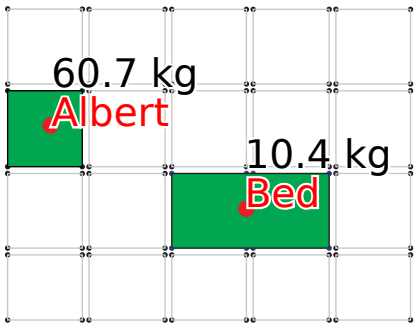
Algorithm 6.1 presents the pseudocode for the exhaustive search of the space of joint blob evolutions. It contains the functions for recursively generating all possible joint blob evolution hypotheses, and for adding them to an ordered list. A comparator allows the solutions to order themselves when adding them to the list of all possible solutions. The list of all possible joint blob evolutions is ordered and bounded in size, keeping only the top evolution hypotheses, which is equivalent to performing *Beam search*.

An *appear* evolution penalizes the weight of the appeared blob, as well as the distance between the new blob and the entry/exit location of the environment. Symmetrically, a *disappear* evolution penalizes the weight of the disappeared blob, and the distance to the exit point. A *split* evolution penalizes the difference between the weight of the parent blob and the total weight of the child blobs. A *merge* evolution penalizes the differences between the total weight of the parent blobs, and the weight of the (unified) child blob. In terms of distance, both *split* and *merge* penalize the euclidean distance between parent and child blobs. However, this distance penalty is considered nil for the parent and child blobs that overlap and occupy the same surface tiles. (e.g. all the split child blobs that are contained within the surface of the parent blob; all the merged parent blobs that are contained within the surface of the unified child blob).

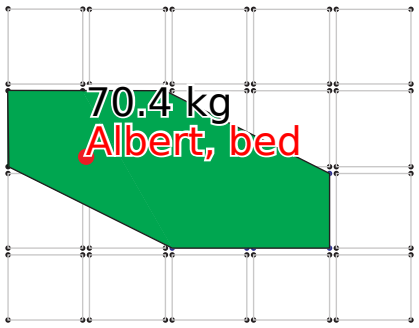
The final score of a hypothesis is obtained by first dividing the distance penalty and the weight

Evolution	**Distance penalty**	**Weight penalty**
Appear or Disappear	$\left(\frac{distanceTo(entrance/exit)}{avg\ COP\ scattering\ noise}\right)^2$	$\left[\frac{weight(blob)}{avg\ pressure\ noise}\right]^2$
Merge	$\sum_{\text{parents}}^{\text{non-overlapping}} \left(\frac{distanceTo(childBlob)}{average\ COP\ scattering\ noise}\right)^2$	$\left[\frac{\left(\sum^{\text{parents}} weight(blob)\right) - weight(childBlob)}{average\ pressure\ noise}\right]^2$
Split	$\sum_{\text{children}}^{\text{non-overlapping}} \left(\frac{distanceTo(parentBlob)}{average\ COP\ scattering\ noise}\right)^2$	$\left[\frac{weight(parentBlob) - \left(\sum^{\text{children}} weight(blob)\right)}{average\ pressure\ noise}\right]^2$

Table 6.1 – Calculation of penalties for each type of blob evolution

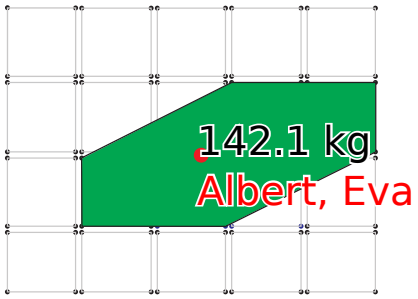


(a) Before the merge. Albert is walking towards the bed.

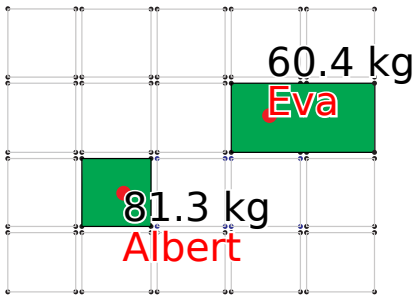


(b) After the merge. Albert and the bed are segmented in a common blob, due to their proximity.

Figure 6.4 – A *Merge* blob evolution, where several blobs unite to form a new common blob. The new *child blob* overlaps with the *parent blobs*.



(a) Before the split. Albert and Eva are segmented in the same blob, due to their proximity.



(b) After the split. Albert walks away from Eva.

Figure 6.5 – A *Split* blob evolution, where a blob splits into two or more new blobs. The *parent blob* overlaps with the new *child blobs* The (noisy) weight information helps to identify how the components of the original blob distributed themselves between the newly appeared blobs after the separation.

penalty by their corresponding average noise values (to have a common unit of measurement for both values), squaring the results (to prefer small errors over large ones), and then summing them up to obtain the final mixed score. The joint blob evolution hypothesis with the lowest penalty is considered to be the most probable one.

The tracking and localisation algorithm proposed in Alg. 6.1 will be evaluated in section 6.3.

6.2.4 Object recognition

Object recognition on load sensing surfaces can be performed by using the weight of objects, or by using their surface of contact with the floor, as suggested in chapter 5. Recognition by weight is trivial when tracking single objects or when performed on high resolution pressure sensors, that can easily segment objects on the floor. However, the problem is less trivial when tracking multiple entities, each with multiple points of support, that interact and perform dynamic activities on noisy, low-resolution sensors.

Background subtraction, connected-component labeling, and blob tracking, described in the previous sections 6.2.2 and 6.2.3, reduce the problem to recognizing the contents of blobs of known weight, which support one or more objects in their entirety. This allows us to model the

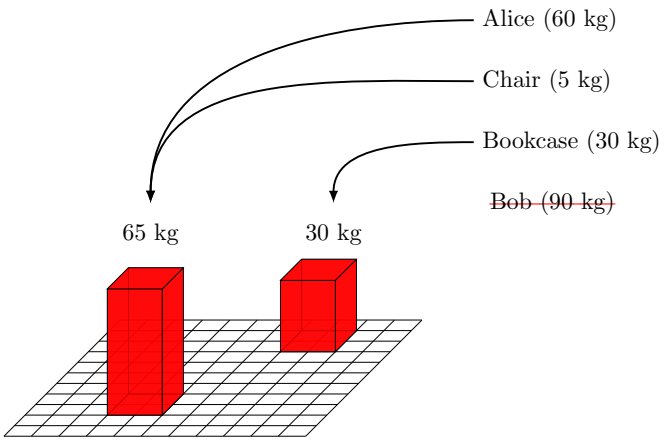


Figure 6.6 – Object recognition modeled as a Multiple Knapsack Problem.

recognition task as an instance of a *Multiple Knapsack Problem* [88], interpreting the weights of detected blobs as knapsacks' volumes, that have to be optimally filled with known objects' weights. This is based on the hypothesis that the average weight of a blob is optimally matched by the weights of the objects it contains (see Fig. 6.6).

This can be formalised as follows:

- $\mathcal{O}$ is the set of known objects;
- $\mathcal{P}(\mathcal{O})$ is the set of all combinations of known objects (it is the power set of $\mathcal{O}$);
- $C = \{blob_1, \ldots, blob_n\}$ is the set of all blobs observed at a given time t. Each blob is defined by its location and weight.

All the possible assignments of objects to blobs are considered and ranked in ascending order, by using the total mismatch in weight between the blobs and the objects assigned to them (see Fig. 6.7). For each hypothetical assignment of objects to blobs, these mismatches are squared and then summed, so as to give preference to small mismatches, rather than large ones. The weight mismatch (or the penalty) of an assignment of objects to blobs is given by:

$$\sum_{id=1}^{\text{total blobs}} \Big(\text{weight}(\text{blob}_{id}) - \text{weight}(\text{contents}\,(\text{blob}_{id})\,)\Big)^2 \tag{6.1}$$

where *contents(blob)* returns the set of objects contained in blob, according to a given assignment.

Algorithm 6.1 – Object tracking algorithm

```
// Generate recursively all possible blob
    evolution hypotheses
GenerateAllPossibleBlobEvolutionHypotheses
   (timeStart, timeEnd)
{
   Extract the blobs present at timeStart.
   Extract the blobs present at timeEnd.
   Instantiate an empty list of joint blob
       evolution hypotheses.
   Create a joint blob evolution structure,
       which will be used to loop through the
       space of solutions.
   Recursively fill this joint blob evolution
       structure, by associating to each
       initial blob a set of its child blobs.
   Recursively generate all possible
       solutions, and add them to the list of
       joint blob evolution hypotheses.
   Sort the list of joint blob evolution
       hypotheses, using the defined tracking
       evaluation function.
   Return this list of joint blob evolution
       hypotheses.
}


// This function associates to each initial
    blob a set of its child blobs
FillEvolutionSolution(
   int initialBlobIDToEvolve,
   int totalInitialBlobs, int totalFinalBlobs,
   long timeStart, long timeEnd,
   Vector<{Vector<Integer>}>
       initialBlobEvolutions,
   List jointBlobEvolutionHypotheses
)
{
   If all blobs from timeStart have been
       assigned blobs from timeEnd, then
       generate a joint blob evolution
       hypothesis.
   Else
      - Attempt all the possible Merge
          evolutions for the current blob from
          timeStart, and call the next
          recursion level on the next blob.
      - Attempt all the possible Split
          evolutions for the current blob from
          timeStart, and call the next
          recursion level on the next blob.
      - Attempt the Constant and Disappear
          evolutions for the current blob from
          timeStart, and call the next
          recursion level on the next blob.
}


// Generate a solution from a pre-filled list
    of blob evolutions.
GenerateABlobEvolutionHypothesis(
   int totalInitialBlobs, int totalFinalBlobs,
   long timeStart, long timeEnd,
   Vector<{Vector<Integer>}>
       initialBlobEvolutions,
   List<JointBlobEvolution-Hypothesis>
       topScoringJointBlobEvolutionHypotheses
   )
{
   Transform the list of correspondences of
       blobs from timeStart to blob from
       timeEnd into a set of blob evolutions,
       each affecting one or more blobs.
   For all the blobs from timeEnd that have
       not been assigned any parent blobs from
       timeStart generate Appear evolutions.
   This set of blob evolutions forms a joint
       blob evolution hypothesis.
   Evaluate the hypothesis, and add it to the
       list of hypotheses.
}
```

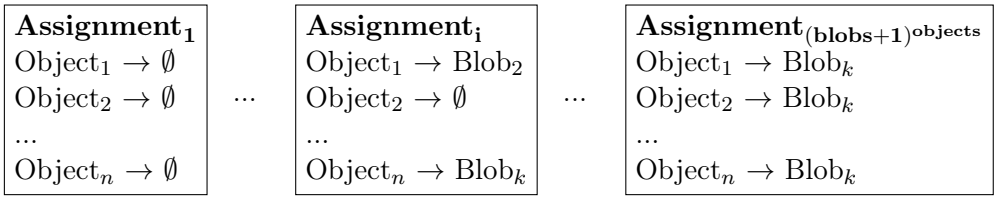


Figure 6.7 – All the possible assignments of known objects to blobs are evaluated and ordered according to how well the blobs are matched in terms of weight by their contents. The assignment having the minimal weight mismatch is considered as most probable.

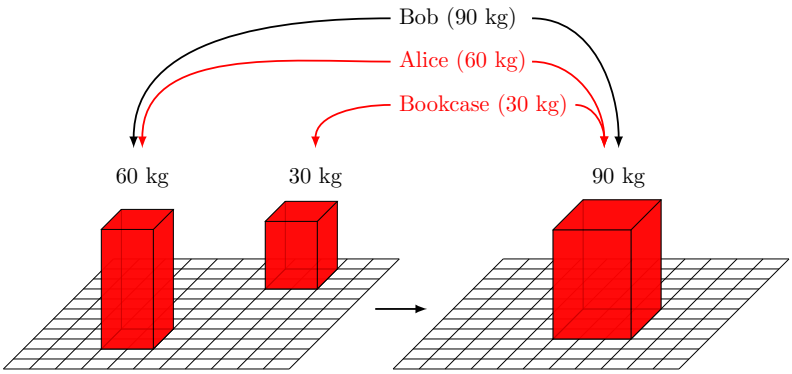


Figure 6.8 – The optimal assignment of objects to blobs cumulates the minimal penalty over time.

The size of the search space, i.e. the number of possible assignments to analyse, is given by:

$$(\text{number of blobs} + 1)^{(\text{number of known objects})} \tag{6.2}$$

As highlighted by eq. 6.2, there is a risk of a combinatorial explosion when performing this exhaustive search. However, most of these hypotheses can be quickly discarded by a *branch and bound* algorithm. In addition, by working only on the localisation hypotheses inferred from the localisation of objects at the previous timestep (as detailed below), the search space gets reduced enough to be dealt with in real time.

Given the contents of blobs in the previous timestep, and given a hypothesis on how the blobs have evolved inside the scene from the previous to the current timestep, we can infer the contents of blobs at the current timestep. However, this requires bootstrapping the knowledge about the contents of blobs at some initial time t_{start}.

As the blobs inside the scene evolve, the candidate recognition solutions will cumulate penalties over time. The candidate solution with the minimal total penalty over time is considered to be the best guess (see Fig. 6.8).

The result of the recognition algorithm is a list of assignments of objects to blobs, ordered according to their cumulated penalties. Intuitively, the assignment having the minimal penalty is considered to be the most probable one.

For probabilistic reasoning algorithms, a measure describing the probability for an assignment of not being the correct solution can be introduced: this is the penalty of the assignment, normalized using the sum of all assignments' penalties (see eq. 6.3).

$$P(\neg Assignment_k) = \frac{\text{Penalty}(\text{Assignment}_k)}{\sum_{id=1}^{\text{total assignments}} \text{Penalty}(\text{Assignment}_{id})} \tag{6.3}$$

6.3 Experiments

We evaluate our approach by running experiments with humans performing daily life activities: doing the morning routine (waking up in the bed, using the toilet, having breakfast), and receiving

(a) The bedroom and living room.

(b) The living room, the bathroom and the kitchen.

Figure 6.9 – The prototype apartment with the tiled sensing floor.

a visitor (opening the door, leading the visitor into the living room, having a chat while seated, eating a cake, leading the visitor to the exit). All scenarii involve multi-object detection, tracking and localisation. The experiments took place in our prototype apartment (Fig. 6.9).

During the whole duration of these activities, the sensing floor had to localize the persons and objects inside the scene. The center of pressure (COP) of a blob was considered as the location of all the objects contained by this blob. An approximation of the ground truth was provided by a Qualisys [9] motion tracking system (tracking error below 1 mm), which recorded the vertical projection of markers placed on objects' approximate centers of mass. Each human had a reflector placed on his waist, so that its vertical projection onto the ground plane would approximately correspond to his COP (fig. 6.10b and 6.11a).

The experimental results are presented as measurements of the localisation precision. These measurements were made only when both localisation data were available: the approximate ground truth given by the motion tracking system, and the localisation provided by the floor. This explains the interruptions in the curves showing the localisation precision.

These error measurements depend on the size of the segmented blobs, in which the objects find themselves. For instance, interacting objects will tipically be segmented into blobs bigger than those containing individual objects. Objects segmented together will be further away from their common center of pressure than an isolated object is from its own center of pressure. Thus, although objects may be imprecisely located inside the segmented blob that contains them, the localisation of regions with obstacles is precise. Therefore, errors on object localisation inside their englobing blobs have a limited impact on robotic navigation and obstacle avoidance tasks, as robots can still identify the zones that have to be circumnavigated.

6.3.1 Morning routine scenario

The *morning routine* scenario involved a person performing a set of daily life activities, such as: sleeping in bed, using the toilet, having breakfast, and leaving the house (see Fig. 6.10). The challenges of this scenario included tracking multiple interacting entities on a low resolution sensor, as well as the presence of ambiguity between objects of similar weight.

The scenario unfolded itself as follows: 8 s — the human enters the scene, 14.5 s – 32 s the human lies on the bed, 35 s – 48 s the human sits on a chair (which acts as a toilet), 50 s — the human grabs a plate and goes towards the chair (near the table), 56 s–65 s the human sits on the chair at the table and has breakfast, 70 s — the human puts the plate back to its original place (passing near the bed), 72 s — the human leaves the scene.

[9] http://www.qualisys.com/

The measured localisation error, obtained using the COP approximation given by the motion tracking system, is expected to be higher than the real localisation error. Unlike robots, humans are flexible, preventing us from calculating an approximation of the COP using the least squares method, that would minimize the overall measurement error (as we did when calculating the baseline precision for the localisation of mobile entities, in Section 3.2.7).

The localisation errors for the human (average error 13 cm) and the bed (average error 19 cm) are shown in Fig. 6.10c. When the person interacts with the bed, the two are segmented together in a single blob, with the COP closer to the heavier human, which explains his better localisation. The localisation error is the biggest at the beginning and end of each interaction (14 s – 32 s lying in bed, 65 s – 70 s passing near the bed), when the two entities begin approaching each other, forming an elongated blob. The localisation error is at its lowest during the close interactions between objects, when the blob regrouping the interacting objects is compact. We observe 15 cm of localisation error for the bed when it is at rest, and a higher error during interactions, which depends on the proximity with the interacting entity and its relative weight. Occasional, short-time errors in the correct assignment of objects to blobs generate localisation errors, seen as spikes in Fig. 6.10d. These errors are due to ambiguities between objects of similar weight. The spikes in the chair localisation error are due to the human proximity, having as effect the segmentation of the two in a single blob. This happened when the human sat on the chair (35 s – 48 s, and the again between 56 s and 65 s).

Fig. 6.10d shows the localisation errors for lightweight objects, such as the chair (5.5 kg) and the dish with the breakfast (4.9 kg). A heavy plate was chosen to overcome the noise threshold of the floor sensor. We observe the same effects as previously described: the localisation error increases in the proximity of humans, due to their segmentation in a common blob, with the COP closer to the human.

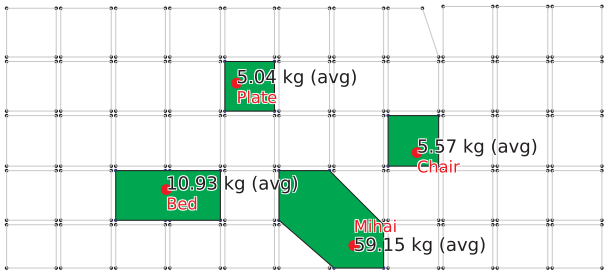
6.3.2 Receiving a visitor scenario

The *receiving a visitor* scenario involved a person hosting someone in his house. The guest would be greeted at the door by the host, enter the living room of the apartment, take a seat, wait for the host to bring something to eat, have a chat with the host, and then leave the house, The challenge was to track and locate multiple interacting persons with a low resultion sensor.

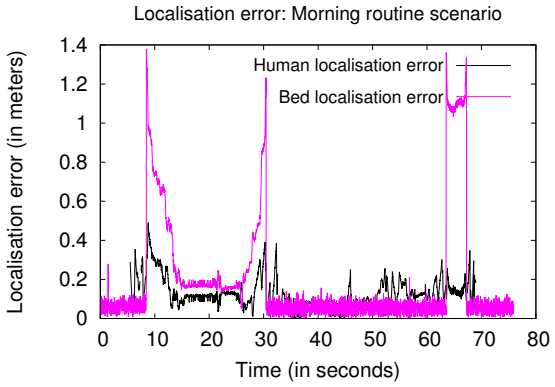
The scenario unfolded as follows: 0-20 s the host walks through the house, walking near chairs; 16.5 s the guest enters the house, sharing a handshake with the host; 22 s — both have a seat; 35 s — the host stands up and takes the plate; 43 s — the host hands the plate to the guest; 48.5 s — the host seats back on his chair; 58 s — the guest stands up and prepares to leave; 60 s — a farewell handshake; 62.5 s — the guest exits the house; 64.5 s — the host follows him suit.

The localisation results are shown in Fig. 6.11c. The average localisation error for the interacting persons is around 20 cm. The drop in the localisation precision occurs when the two persons walk or stand nearby, occupying a contiguous space in terms of tiles, which prevents them from being segmented separately. These close interactions occur when passing the plate to the guest (at 43 s), and shaking hands (at 16 s during entrance and at 60 s while leaving). Again, the measured localisation error for humans is expected to be higher than the real localisation error, as we could not approximate the position of the human's COP with the least squares method, as we did it with a rigid robot when calculating the baseline of the floor's localisation error in Section 3.2.7.

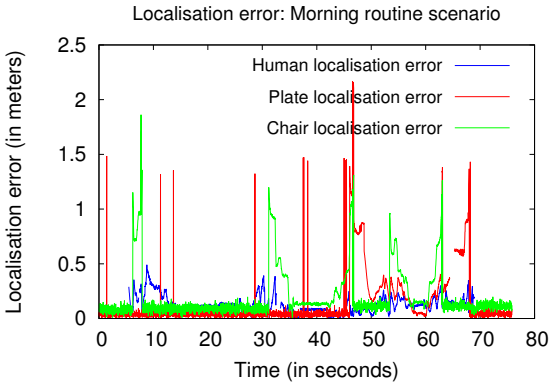
(a) An image from the *Morning routine* scenario.



(b) A frame from the *Morning routine* scenario, as seen from the floor's perspective.



(c) Localisation error for the *Morning routine* scenario. During each interaction between the person and the bed, the center of pressure is located between the interacting entities, closer to the heaviest one (the human, in this case). The spikes between 10-30s, and between 62-68s are caused by the human approaching and distancing from the bed during the interactions.



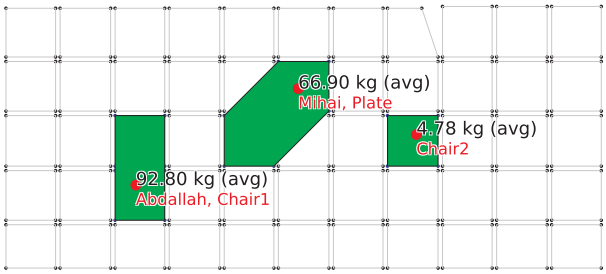
(d) Localisation error for the lightweight objects in the *Morning routine* scenario. Occasional, short-time errors in the correct assignment of objects to blobs generate the thin spikes in the localisation error. These are due to ambiguities between objects of similar weight. The spikes in the chair localisation error are due to the human proximity, having as effect the segmentation of the two in a single blob.

Figure 6.10 – The *Morning routine* scenario

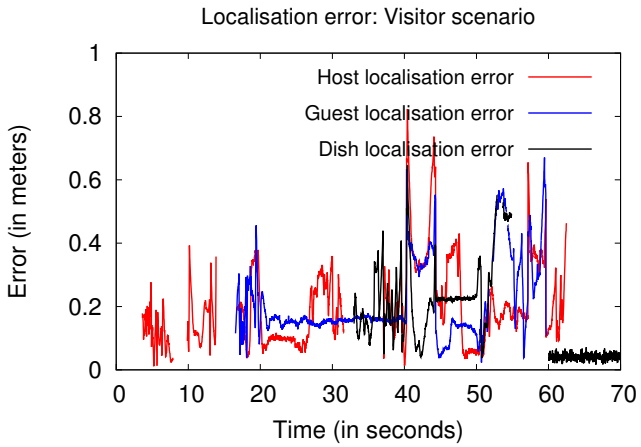
6.4 Conclusion and perspectives

This chapter presented a technique for detecting, tracking and recognising objects on load-sensing floors, using objects' weight as discriminative feature. The proposed object segmentation algorithm is a variation of *connected-component labeling*, inspired by the computer vision community, with the additional property of having entire objects segmented into blobs. This allows the interpretation of spread loads, when objects span several tiles on a modular floor. The proposed tracking algorithm considers the different ways in which blobs can interact, identifying the most probable hypotheses for the way the blobs have evolved between two timesteps. This allows to infer the objects contained in the segmented blobs, given their contents at the previous timestep, and given a hypothesis on the evolution of blobs. The resulting possible assignments of objects to blobs are ranked by the mismatch between the weight of blobs and of objects assigned to them. This is reminiscent of the *multiple knapsack problem*, with blobs acting as containers that have to be optimally filled with known objects, identifying the optimal solution using the *Least Squares* method.

(a) An image from the *Visitor* scenario, as seen from a camera's perspective.



(b) A frame from the *Visitor* scenario, as seen from the floor's perspective.



(c) Localisation error for the scenario with a person hosting a visitor in his house. The drop in the localisation precision occurs when the two persons walk or stand nearby, occupying a contiguous space in terms of tiles, which prevents them from being segmented separately. The average human localisation error is 20 cm.

Figure 6.11 – The *Visitor* scenario

The presented algorithm works independently of the floor resolution (i.e. density of sensors per m^2, size of the floor tiles). However, the bigger the tiles are (the lesser the floor-image resolution is), the coarser will be the results of the algorithm performing object detection and segmentation. Coarser object segmentation results lead to more ambiguity in object recognition and localisation. Therefore, it would be interesting to have a prototype with a higher sensor density (smaller tiles in our case), as well as less noisy sensors.

The whole localisation algorithm was evaluated in experiments with humans performing daily life activities: executing the morning routine, and receiving a visitor. Challenges included the segmentation, tracking and recognition of multiple interacting entities using a low resolution sensor, as well as the disambiguation between combinations of objects of similar weight. The average error for human localisation was approximately 20 cm. The result of this algorithm can be modelled as a probability distribution over all possible assignments of objects to the blobs detected on the floor. This allows for easy integration of this algorithm into a multi-modal object recognition architecture. This technique can boost recognition when used complementarily with algorithms that extract features from the human gait, but can also serve as a gracefully degraded recognition mode whenever these fail.

The main impact of this work is allowing the detection, tracking and recognition of humans

inside their habitat for ambient intelligence applications. These can include the targeted collection of biometrical data for medical analysis and continuous diagnosis of human's state of health, and the generation of occupancy maps for robots functioning in the same environment.

6.4.1 Future work

Future work will include fine-grained tracking, obtained by assigning each detected object to a separate layer. This should allow to continuously update the set of objects composing the background, and would consequently improve segmentation. We are also working on labeling the interactions between humans and objects, which can be roughly observed using this technique. We also plan to use this data to generate logs detailing the activities performed by a person during the day: how many times a person got out of bed, how many steps did she make, how many persons are there in the room, etc. These activity plots are useful in hospitals and retirement homes, as they allow to trace the overall health state of a patient, or to detect situations when the monitoring system should be turned off to respect human privacy.

It would also be interesting to distribute the computation of the object tracking and localisation functions presented in this chapter, making use of the processing units on board the tiles. This would allow the algorithm to scale well with an increase of the surface of the floor.

This chapter concludes Part II, which shows some capabilities of the load-sensing floor as an individual sensor. In Part III, we will see how the floor can be combined with the robotic actuators of an environment with embedded ambient intelligence. In particular, we will demonstrate how the sensing floor can provide support in navigation and exploration to autonomous robots evolving on it.

Part III

Integration of sensing floors with personal assistant robots

7

Providing roadmaps for autonomous robotic navigation

Contents

Integrated into an ambient intelligence system with robotic actuators, a floor pressure sensor can provide safe navigation routes for robots by creating 2D occupancy maps of the environment. It can also track and locate robots inside this map, providing an alternative to Simultaneous Localization and Mapping (SLAM). This gives room to the simplification of robots' structural design, potentially rendering some sensors redundant. This non-intrusive alternative to SLAM is explored in this chapter.

7.1 Introduction

In order to navigate through an environment, a robot has to learn a representation of the world [163] and localize itself within it [165], i.e. solve the SLAM problem [41, 93]. As precise localisation is important for matching and merging new map scans with the existing map with minimal errors, a high-precision sensor is usually employed, such as a light detection and ranging (LiDAR) sensor. This precision equipment can make up for half the cost of a small robot. Externalising its functions could allow to significantly reduce the manufacturing costs of robots. The computing effort to calculate solutions for the SLAM problem can be done on-board by mobile robots [163], or off-board, exploiting the surrounding environment's processing capabilities [15, 84, 167].

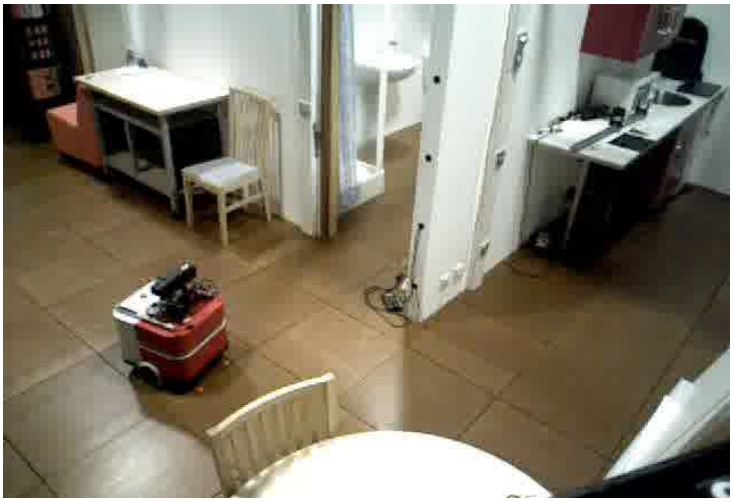


Figure 7.1 – An autonomous robot navigating on a pressure sensing floor.

This chapter introduces an environmentally-computed navigation service, provided by an environment equipped with a load-sensing floor. It uses the information given by sensing and communicating floor tiles, in order to generate a maximum clearance roadmap (also known as a Voronoi diagram). These Voronoi diagrams can then be used by robots to plan their navigation through the environment [34], using the shortest of safe paths [51], smoothing path-planning algorithms [99], and multi-agent path planning [149]. Our off-board navigation service provides the core functionalities of localisation and mapping for robotic navigation, supplying a metric map of the environment and localisation within it.

The navigation service relies on two modules: a first one dedicated to occupancy computation and client localisation, and a second module for computing a maximum clearance roadmap from the occupancy map and providing it to the client entity. The computation of the Voronoi diagram is impeded by the desynchronisation between the tiles composing the floor. Synchronised Voronoi diagram computation algorithms, either parallel or sequential, process the cells composing the environment in a predefined order. Under this assumption, if a cell fails to communicate its state, the computation may fail to converge to the correct result. We overcome this aspect by using an asynchronous, distributed algorithm for computing Voronoi maps on 2-dimensional grids, introduced by Kaldé *et al.* , our research team colleagues, in [67]. This approach for computing and maintaining safest navigation paths is self-organizing, which enhances robustness to noise, adaptivity to changes, and scalability to larger sensing networks.

Given that it perceives everything from the ground plane, the floor does not suffer from occlusion, as compared to on-board sensors that the robot may have. It can recognise users and objects by the pressure they exert on the floor, and keep a larger distance from humans when navigating. Our goal is to facilitate navigation for autonomous machines, like mobile robots, autonomous wheelchairs, and even for blind handicapped humans wearing audio guidance devices.

By externalising core functionalities as localisation and mapping into a service provided by the environment, we open the doors for low-cost mobile robots that have limited on-board sensing and processing capabilities [12, 68]. This allows a robot to ship with simple sensors, actuators, enough computation power dedicated to its specific task, and a communication module for querying services provided by the environment.

The remainder of this chapter is organised as follows. Section 7.2 presents the related work on

sensor networks for roadmap computation. Section 7.3 provides the details of our methodology, explaining each element of our data workflow, from the raw load-sensors' output to the final navigation roadmap. In Section 7.4, we define our experimental protocol and discuss a real-life scenario implemented on a load-sensing floor. We conclude this chapter with some remarks regarding planning using a roadmap computed by a heterogeneous sensor network, and provide ideas for future work. This chapter is based on a paper in progress [10].

7.2 Related work

This section provides an overview of ambient intelligence systems, which offer services that exploit their sensor networks. According to a 2010 survey by Huang and Gartner [62] on mobile indoor navigation systems offering location based services, all such systems provided positioning, route calculation and route communication services.

In the sensor network community, an early application to robotic navigation was proposed by Li *et al.* [84], in which spatially localized sensors detected hazardous situations around them, and then broadcasted this information through the network using flooding techniques. Shortest and safest paths were distributively computed using distances measured in hop counts, and potential field techniques inspired from robotics, respectively. Unfortunately, the algorithm did not scale well due to prohibitive communication costs.

In the robotic community, Batalin *et al.* [15] worked with a network that served as the sensing, computing, and communicating medium for the robots, which provided only actuation. The path planning problem was formulated as a Markov Decision Process. They proposed a probabilistic navigation field, computed by a distributed value iteration algorithm over the network, which could find near-optimal safe paths. This work assumed that states and transitions are computed *a priori* to maximize the expected utility for reaching a goal state.

Verma *et al.* [174] investigated credit-based navigation fields for hybrid sensor networks, where sensors on mobile platforms were guided by static sensors to detected events. The assignment of mobile sensing units to events that had to be investigated was done according to criteria of distance to the event, the power of each mobile sensing unit, and the evolution of visual coverage.

Buragohain *et al.* [26] proposed a computation of near-optimal safe paths (with minimal exposure to danger, or shortest feasible paths). Paths were planned on demand using a breadth-first search in a subgraph of the environment. This work was inspired by mesh generation algorithms from the computer graphics community. Uniform and adaptive skeleton based roadmaps were proposed. Hovewer, no moving obstacles were considered.

Similarly, Alankus *et al.* [5] introduced an adaptive embedded roadmap for navigation, where paths are evaluated according to their length and maximum danger level. Reduced communication and path planning costs were achieved thanks to look-up tables storing up-to-date information.

Probabilistic frameworks for computing roadmaps for path planning have also been investigated. Yao and Gupta [189] proposed a distributed path-planning algorithm where each node creates a local probabilistic roadmap and computes a part of the plan using Dijkstra's algorithm.

Voronoi-based roadmaps were studied by Wang *et al.* [181]. They stated that selectively updating the roadmap structure can also reduce the communication overhead while providing safest routes for navigation at the expense of path length.

We follow the previously mentioned work, calculating adaptive roadmaps based on Voronoi diagrams. The innovation in our approach stems from using load-sensing floors to detect and track objects, and to construct occupancy maps. We then compute a Voronoi diagram which

serves as roadmap, with passages equidistant to obstacles' borders, using the algorithm presented by Kaldé *et al.* in [67]. This algorithm can adapt to changes in the occupancy map, without recalculating the entire Voronoi diagram. Shortest and safest paths can then be computed on these roadmaps, using any state-of-the-art planner.

7.2.1 Sensing floors for robotic navigation

Although localisation of objects using sensing floors has been an active research topic since the 1990's, as described in Chapter 6, Section 6.1, no effort was made to exploit the ability of floors to generate occupancy maps for robotic navigation.

In the closest work of such type, Khaliq and Saffiotti [70] used a floor with embedded RFID tags to guide robots to pre-defined goals in the environment, using a stigmergic approach. However, the distances to obstacles had to be hard-coded in the RFID nodes, disallowing any modification of the environment, as it would provoke incorrect path calculations.

In comparison, we show how load-sensing floors can continually update their occupancy maps, by using the information about the weight located on the floor. We need not emphasize that up-to-date maps are critical for safe navigation.

For the detection and localisation of objects evolving on the floor, we used techniques stemming from computer vision, such as *background subtraction* and *connected-component labeling*. These techniques can be almost readily applied to pressure-images, as they are analogous by their construction to photo-images (the former represent amounts of pressure, while the latter represent amounts of light). Object tracking was performed using an exhaustive search method, minimizing the discrepancy in weight and position between correlated objects in two sequential time-frames, as it was described in Chapter 6.

7.2.2 Distributed computing of the discrete Voronoi diagram

A *Voronoi diagram* identifies the regions that are equidistant from a given set of obstacles, according to some distance metric (e.g. Euclidean, Manhattan, or Minkowski distance) They are used in robotics to identify the safest paths from an obstacle-avoidance standpoint.

Two techniques exist for computing Voronoi diagrams. The first comes from the field of Cellular Automata, where these diagrams are used to compute the Medial Axis Transform using the *grassfire transform* [21]. These Voronoi diagrams were then employed for path planning [168, 2, 1]. The second technique comes from the field of Image Processing, where Voronoi diagrams are computed using distance transforms [132, 46] and sequential rules applied locally on each cell.

Kaldé *et al.* [67] developed a distributed computation of Voronoi diagrams using an asynchronous cellular automaton, employing neighbourhood relationships for cells inside a Minkowski 1-norm distance. This chapter continues their line of thought, combining the obstacle-detection service provided by a discretized load-sensing floor with the continuous distributed computation of Voronoi diagrams for robotic navigation. In the rest of this chapter, we will refer to Voronoi diagrams as defined in [103].

7.3 Methodology

This section presents the components of our roadmap extraction service. The problem is separated in two parts: occupancy grid calculation and extraction of navigation routes, which allows

a modular design. We then introduce the methods employed for the generation of the occupancy map, as well as for the calculation of the maximum clearance roadmap.

7.3.1 Environmental data flow

The data flow employed in our contribution is illustrated in Fig. 7.2. The raw data perceived by the sensor network serves to detect the objects occupying the environment. The generated occupation grid is then used to calculate navigation routes. The computation of navigation routes is done by letting the tiles composing the sensing floor to distributively calculate a Voronoi diagram separating the detected obstacles. These two processes are detailed in the following sections.

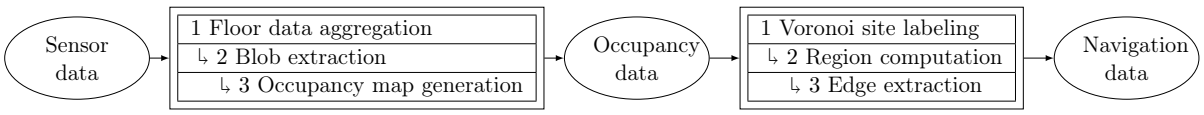


Figure 7.2 – Data Workflow diagram

7.3.2 Detection of obstacles and mobile objects using the sensing floor

We implement our navigation service on a network of load sensors embedded in the floor of an appartment, presented ealier in chapter 3. We use the object detection and tracking capabilities of the floor, described in chapter 6, to identify the occupation state of each tile. This allows to generate at each timestep a binary image describing the occupancy state of the entire floor. In addition, we label the type of objects occupying the floor, discriminating between static and mobile objects (see Fig. 7.3). This is useful for motion planning, as it allows to leave more space around mobile entities when navigating.

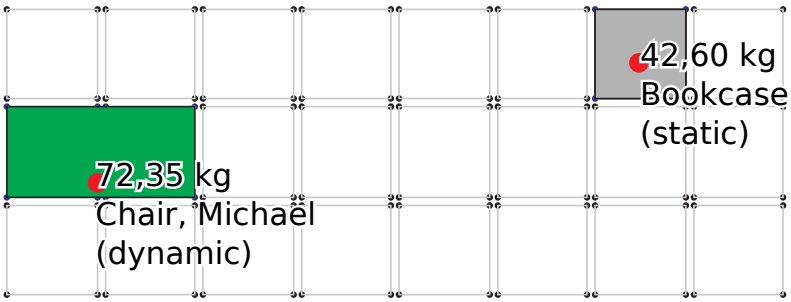


Figure 7.3 – Localising and recognising objects: discriminating between static and mobile entities.

7.3.3 Distributed line Voronoi diagram computation

Although computing Voronoi diagrams is not novel, computing them using information provided by a sensing floor is innovative. The difficulty lies in calculating the diagram asynchronously, as the tiles of the sensing floor do not guarantee synchronisation. The tiles can fail to send data, and the data packets can be delayed by the network and thus be unavailable for a centralised computation of the Voronoi diagram. Calculating and constructing the diagram in a distributed manner allows to avoid the inconvenients of a less robust, centralised system. Kaldé *et al.*

introduced in [67] an algorithm that can distributively compute and maintain an up-to-date Line Voronoi Diagram in a discretised environment with cells aware of their occupancy state. This is the solution we have implemented on our sensing floor.

The algorithm consists of three steps. First, during *Voronoi site labeling*, it labels the edges of obstacles. Then it computes the *Voronoi regions* by propagating the labels of each obstacle edge through the environment, similar to a forest fire propagation. Finally, the cells whose neighbours have a different set of closest obstacles identify themselves as frontiers between Voronoi regions. These three steps are detailed below.

Voronoi site labeling

The component labeling procedure identifies the horizontal, vertical and diagonal edges of an obstacle. Cells belonging to the same edge have to obtain a common identifier. To achieve this, each cell detects its edge pattern among the possible mask classes presented in Fig. 7.4. Then, the cells check their neighbours' states, to detect conflicting situations, in which case a standard priority resolution is applied.

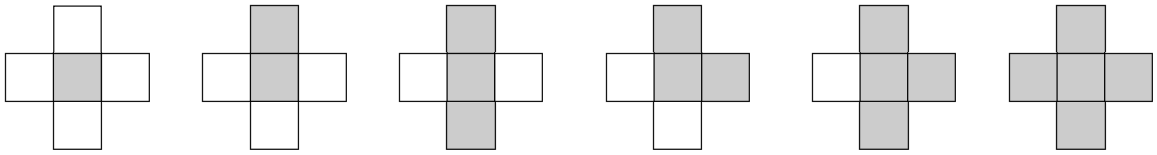


Figure 7.4 – Six classes of site patterns, together with the number of patterns within the class. The dark cells are cells that have identified themselves as *occupied*, while the white cells have identified themselves as *free* (due to the absence of pressure on the tile).

Voronoi region computation

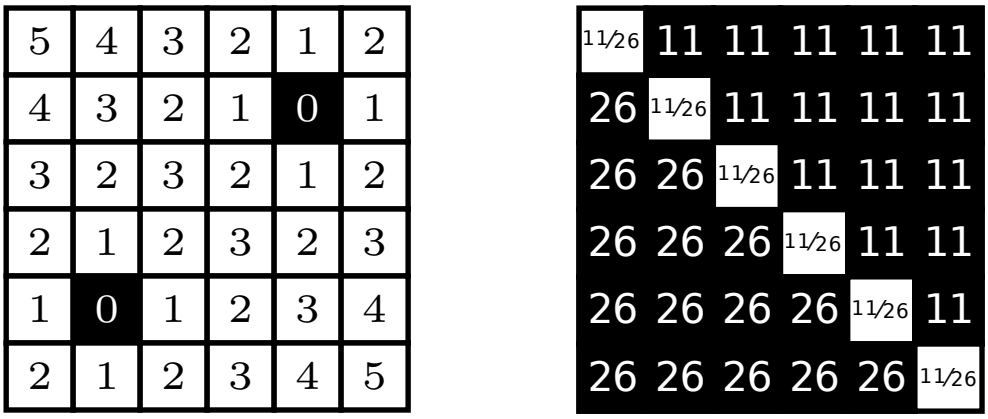
To compute the Voronoi regions, we propagate information about the distance to the sites (obstacles) through a so-called informative gradient. This gradient contains information about the name of the closest site and the distance to it. This allows cells to determine the Voronoi region to which they belong. Each cell updates its distance to the closest obstacle according to the formula:

$$\text{Distance to closest obstacle} = \begin{cases} 0, \text{ if the cell is declared as } \textit{occupied} \\ min(\text{neighbors' distance to closest obstacle}) + 1, \text{ otherwise.} \end{cases}$$

Together with the distance to the closest obstacle, the cells transmit the identifier of the closest obstacle. Whenever there are multiple obstacles at minimal distance, their identifiers are merged into a set. The following rule defines how the identifiers of closest obstacles are propagated:

$$\text{Closest obstacle identifier} = \begin{cases} Id_{cell}, \text{ if the cell is declared as } \textit{occupied} \\ \{\text{set of Ids of closest obstacles}\}, \text{ otherwise.} \end{cases}$$

Fig. 7.6 exemplifies the propagation of the gradient containing the distance to the closest obstacle and its identifier.



(a) The gradient of distances to the closest obstacle. The obstacles are colored in black. The numbers indicate the distance to the closest black cell, using a Von Neumann neighbourhood.

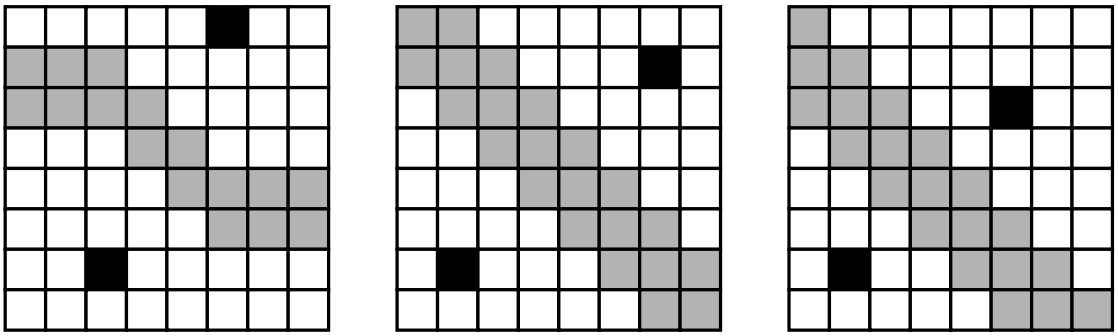
(b) The result of the propagation of obstacle identifiers over the gradient of distances to the closest obstacle. All the cells that share the same set of identifiers of closest obstacles are part of the same Voronoi region. The frontiers between the regions are formed by cells having neighbours with different sets of identifiers.

Figure 7.5 – The propagation of the gradient with the distances to the closest obstacles and their identifiers. Source: Kaldé *et al.* [67]

Voronoi boundaries extraction

The boundaries of Voronoi regions are formed by the empty cells, whose closest obstacles' identifiers are different from their neighbours' closest obstacles' identifiers. Thus, an empty cell possessing a different identifier from at least one of its neighbours will declare itself as a boundary cell. The boundary cells are at the maximum distance from the closest obstacles, and are therefore the safest ones for navigation. The function below defines when a cell considers itself as being part of a boundary between Voronoi regions:

$$\text{Is this cell a boundary cell?} \begin{cases} 1, & \text{if a neighbour's set of closest obstacles' Ids differs from yours} \\ 0, & \text{otherwise.} \end{cases}$$

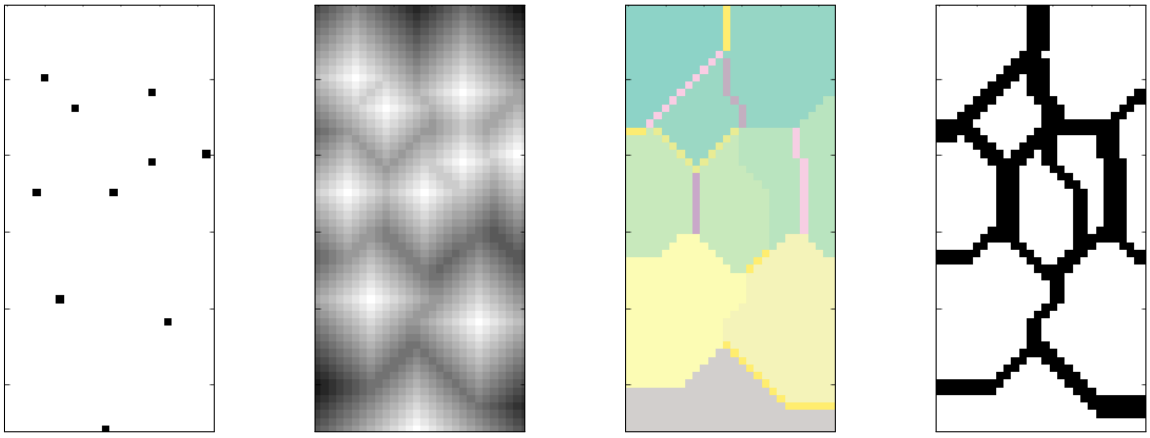


(a) (b) (c)

Figure 7.6 – Samples of calculated thick bisectors (in grey) for grids with obstacles (in black). Source: Kaldé *et al.* [67]

Roadmap computation example

We can now compute a *Voronoi Diagram* in a distributed manner, in an asynchronous grid of cells, as in the case of a tiled load-sensing floor. Fig. 7.7 provides an example of the entire process for calculating a roadmap of the safest routes. The obstacles, that are the sites in the Voronoi diagrams, are represented as black cells in Fig. 7.7a. The gradient of distance to the closest obstacle, calculated in a distributed fashion, is shown in Fig. 7.7b. The identified Voronoi regions are shown in Fig. 7.7c, where cells belonging to the same region share the same color. Finally, the computed boundaries between the regions, that form the safest routes, are shown in Fig. 7.7d.



(a) Map with obstacles shown as black dots.

(b) The corresponding gradient of distance to closest obstacle, in grayscale.

(c) The extracted Voronoi regions.

(d) The boundaries between Voronoi regions are the safest routes.

Figure 7.7 – The extraction of a discrete Voronoi diagram

Once the Voronoi diagram of the environment has been computed, it can be transmitted to robots, that can use it for planning their navigation through the environment [34]. Shortest paths can be identified over the safest routes [51], that can also be smoothed for robots with limitations on the minimal turning radius [99]. In dynamic, crowded environments and in settings with multiple moving agents, multi-agent path plannign algorithms can be used [149].

7.4 Proof of concept

7.4.1 Equipment

The described roadmap-providing service was implemented on the Inria tiled load-sensing floor, presented in chapter 3. Communication between the occupancy map and navigation map modules, described in Fig. 7.2, was implemented using ROS middleware [117]. Occupancy data is transmitted in string format, which contains a JavaScript Object Notation (JSON) structure, making the messages easy to parse using standard tools. The resulting navigation maps are published in a corresponding ROS topic. Clients communicate with the map service providing the maps using ROS via a wireless connection.

7.4.2 Experimental scenario

The experimental scenario involved a person walking through the apartment equipped with a sensing floor, inside which several static objects were placed: two chairs and a bookcase (see Fig. 7.8a). The system first detects and identifies the entities present in the scene (e.g., objects, humans), and labels them accordingly (Fig. 7.8b). It then creates an occupancy map, indicating which obstacles are static and which are mobile. This information is then used to calculate the Voronoi diagram, which provides the roadmap with the safest navigation routes. This roadmap can be finally transmitted to the clients of the system, together with information about their location on the map. Figure 7.8c presents the roadmaps generated in our scenario, while the person was walking through the room.

Recognising and thus differentiating between static and mobile obstacles is useful for selectively updating the maximum clearance roadmap only around active entities in the scene. This service also reduces the planning complexity for the client entities, as instead of working on a full-scale map of the environment they are limited to the roadmap with the safest routes.

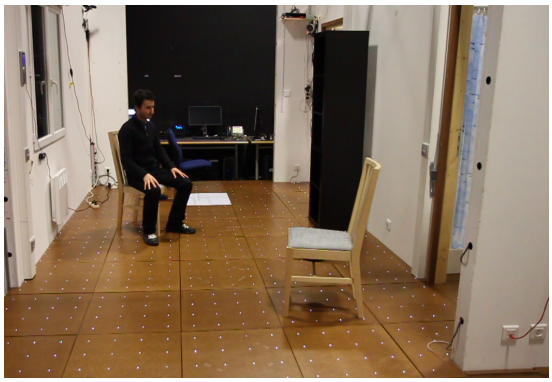
7.5 Conclusion and perspectives

This chapter introduced a service offered by a sensing environment, which provides its clients up-to-date roadmaps of the safest routes in the environment. It is capable of tracking and recognizing the entities that evolve inside its sensing perimeter. Such a navigation aid is useful for autonomous robots with limited on-board sensing or computing capabilities. The Voronoi diagrams can be used for planning the navigation through the environment. The ability of the floor to generate an occupancy map, together with its capacity to locate the clients inside it, could ultimately replace the SLAM module typically required by autonomous robots. This implies that robots with a simpler design can be deployed in environments that provide such navigation services.

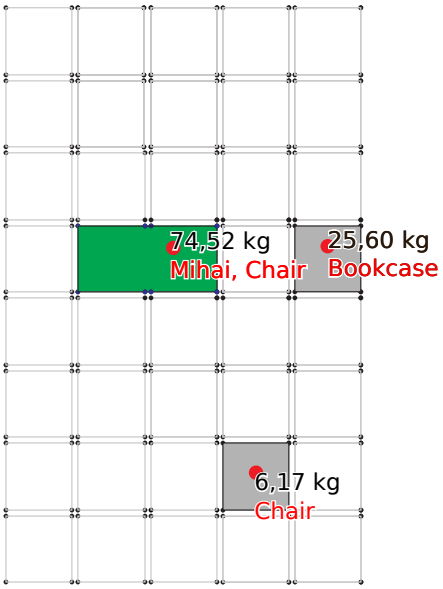
From a practical standpoint, it would be interesting to provide a fully distributed service, by decentralising the computation of the occupancy map, as its constituent algorithm for object segmentation is currently centralised. Rendering the system real-time is another feasible enigneering task.

Our long term perspective is to explore the limits of the ubiquitous computing paradigm, aggregating data from heterogeneous static and moving sensors for the construction of distributed services. More precisely, we plan to integrate the load-sensing floor with depth cameras, audio sensors (microphone grids), proximity sensors, detectors of discrete events (opening/closing of doors), etc. Ultimately, the sensor fusion problem will force the development of new frameworks for the aggregation of heterogeneous sensing data.

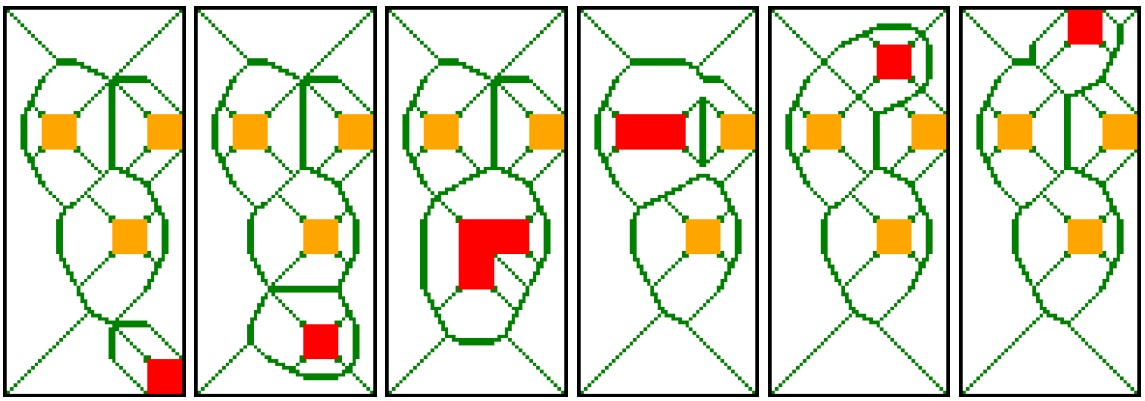
In the next chapter, we will further develop the concept of a floor providing aid to mobile robots, by showing how it can support stigmergic algorithms for environment exploration.



(a) Setting of the dynamic experience, with a human present in the scene.



(b) Occupancy data obtained by the load sensing floor. Weight-based object recognition allows to identify the type of each obstacle.



(c) The Voronoi diagrams extracted while a person entered the scene through the bottom-right corner, sat on a chair, and left the scene through its top-right side. Dynamic and static obstacles are represented as red and orange squares, respectively. The safest navigation routes are shown in green.

Figure 7.8 – The dynamic case experience. The low resolution of the Voronoi line diagram is due to the low density of sensors in the tiled sensing floor.

8

The floor environment as provider of stigmergy for robotic bio-inspired algorithms

Contents

In the previous chapter we have seen how sensing floors can help autonomous robots navigate through the environment, by providing them with roadmaps and localisation. In this chapter, we will explore how environments that are able to store information can help autonomous robotic exploration, using the example of stigmeric algorithms for the distributed exploration of structured environments.

We will present the problem of exploration, and a multi-agent stigmergic exploration algorithm called Brick&Mortar [44], which stands out by allowing agents to distributively identify the end of exploration. We will then introduce the upgrades we brought to this algorithm, resulting in the Brick&Mortar Improved (BMI) and Brick&Mortar Improved with long range vision (BMILRV) algorithms. The former solves the problem of gathering the agents once the task is done, by adding the support for a post-exploration meeting point. It also accelerates the exploration time by improving the mutual exclusion performed by the agents exploring and marking the environment. The latter algorithm generalizes the tabu-list exploration approach to agents with arbitrary vision range, that view the world from a robot-centered perspective, as compared to the top-down perspective previously used in the literature. We will conclude this chapter with prospects for future work on distributed exploration algorithms. This chapter is based on two

publications [6, 7] presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2013 and IROS 2015, respectively.

8.1 Distributed exploration of unknown environments

Exploration of unknown environments is an important field of research for autonomous robotics. Its applications include reconnaissance, search and rescue missions, planetary exploration, as well as exploration of buildings and robotic navigation for home automation. It is also applied for web indexing spiders. Exploration and mapping scenarios usually involve three tasks:

1. Explore and perceive all the space,
2. Identify the end of exploration,
3. Gather explorers at a rendezvous point (e.g. entry/exit point or evacuation point).

Multi-agent exploration of an environment can be formally described as follows: a set of agents explore a graph $Graph$, starting from node $Node_{start}$. The graph known to agents, called $Graph_{agents}$, is expanded through actions of type $navigateTo(node)$. Every time an agent sees a new portion of the graph, this portion is added to the known graph $Graph_{agents}$.

$$
\begin{aligned}
Graph &= \{Nodes, Links\} \\
Graph_{agents} &= \{Node_{start}, Links_{start}\} \\
Agents &= \{a_1, \ldots, a_n\} \\
Actions &= \begin{cases} navigateTo(node) \\ mark(node, markType) \end{cases}
\end{aligned}
$$

Exploration ends when there are no more nodes to expand. When a global map of the environment is available, this end condition can be checked by looking if all the nodes have been expanded. However, a different solution is required when agents have only a partial knowledge of the map.

8.2 Exploration of unknown environments: state of the art

Two main families of algorithms exist today for exploration of unknown environments: (1) frontier-based algorithms, and (2) ant-algorithms.

Frontier-based algorithms guide the exploration by pushing the agents towards the boundaries between the explored space and the space yet unexplored [188, 143, 28, 27, 17]. The exploration ends when all the space was discovered, implying that there are no more frontiers to move to (see Fig. 8.1). The navigation to these frontiers implies planning on the whole map of the enviornment. In case of large maps, this planning may be computationally expensive.

Ant-algorithms are distributed, i.e. multi-agent algorithms, that exploit the capacity of the environment to store information. They use a shared map, on which agents lay traces that guide their exploration. For instance, these traces may allow agents to do a gradient-descent exploration of the environment [180, 74]. Compared to frontier-based algorithms, where agents plan their navigation using their entire knowledge of the map, ant agents reason only on the locally visible fragment of the map. This reduces the complexity of navigation planning. Agents don't have to know their position, and they use no inter-agent communication, except for the marks left in the environment. They can adapt to unknown environments, and they scale well

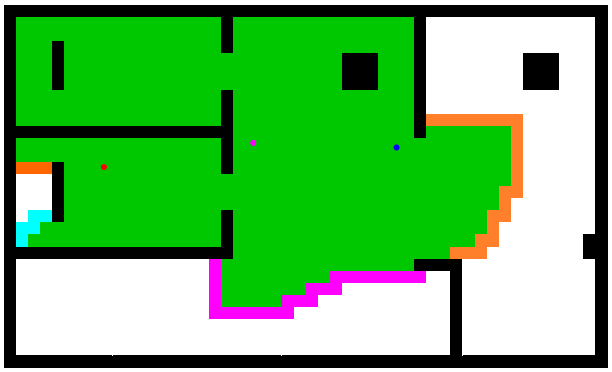
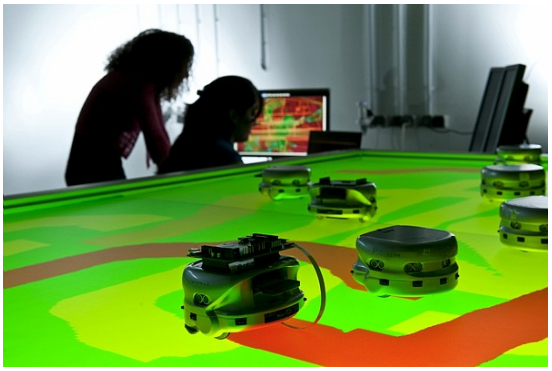


Figure 8.1 – Frontier exploration example. Frontiers between unexplored and explored regions are highlighted in color. Agents, shown here as dots, explore by navigating towards these frontiers. Exploration ends when there are no more frontiers and all the environment is explored.

with an increasing number of agents. The implementations mentioned in the literature make use of thermal trails [134, 22], alcohol trails [140], odour trails [135, 133, 114]) and ink markings [155].

Although ant-algorithms were longtime considered non-realistic for robotic implementations, the emergence of environments with embedded sensor networks shifted this point of view (see Fig. 8.2). Load-sensing floors with embedded processing units, like the floor presented in Chapter 3 (see Fig. 8.2a), can store and dissipate artificial pheromones, and could one day guide cleaning robots through the home. Interactive screen surfaces can be used in a somewhat similar manner (see Fig. 8.2b).

(a) A PekeeII robot on a tiled load-sensing floor with embedded processing units [107]



(b) Khepera III robots performing a foraging task on an interactive table [144]. The intensities of pheromone traces are color coded using a red gradient.

Figure 8.2 – Platforms capable of supporting pheromone traces.

Unfortunately, most distributed algorithms proposed in the literature cannot identify the achievement of full exploration of the environment. The only exception known to us is a tabu-list algorithm, called Brick&Mortar, which is presented in the next section.

8.3 The tabu-list approach for exploration: Brick&Mortar

Brick&Mortar is an ant algorithm introduced by Ferranti *et al.* [44], that at its time was the only one (to our knowledge) capable of identifying the completion of a multi-agent exploration. It views the environment as a graph, through which agents navigate using a taboo-list approach [54]. Agents mark the explored nodes of the graph as closed for access, while still keeping the open parts of the graph connected. This prevents agents from going to regions of the graph that were already explored, and concentrates them in the remaining non-explored parts of the graph. By continually reducing the region of the graph inside which they can navigate, agents get grouped together in the remaining part of the graph. Ultimately, they meet in the last available node, which is the rendezvous point. When this last node is closed, the exploration is considered complete. Gathering all the agents in this node ensures that they are all informed about the end of exploration by looking at the state of the open graph.

In its representation of the environment, Brick&Mortar uses 4 types of nodes: *unexplored* nodes, *explored* nodes (visited nodes which connect the unexplored regions), *closed* nodes (visited nodes that will be avoided during further exploration of the environment)[10] and *walls* (impenetrable nodes).

$$\text{Taboo} = \blacksquare \text{ explored and closed nodes}$$
$$\overline{\text{Taboo}} = \blacksquare \text{ explored, not yet closed nodes}$$
$$\text{Unexplored} = \square \text{ unexplored nodes}$$

The graph to explore is thereby composed of the explored regions, the non-explored regions and the cells linking them : $Graph_{agents} = \text{Taboo}_t \bigcup \text{Unexplored}_t \text{ nodes} \bigcup \overline{\text{Taboo}}_t$ The execution of the algorithm takes place in 2 steps: (1) agents navigate to the next cell, and (2) agents decide how to mark the current cell.

During the exploration of the environment, agents give preference to *unexplored* cells over *explored* ones. If surrounded by *explored* cells, an agent will navigate towards the least-visited cell (a heuristic for dispersing agents known as *Node Counting* in the literature). Access to *closed cells* is forbidden.

When marking the environment, a cell can only be *closed* when doing so does not break the connectivity of the graph portion visible by the agent. This guarantees that all the *non-tabu* regions are connected (that the graph is never cut into separated, disconnected pieces). It also has the emergent effect of keeping all the exploring agents inside the open part of the graph. Thus, as the size of the *non-closed* graph reduces to zero, all the agents will concentrate in a single point, given the assumption that multiple robots can physically share a cell. Therefore, agents will be able to detect when the exploration of the environment has finished, by checking if all the surrounding cells are marked as *closed*. A sample execution of the Brick&Mortar algorithm is shown in Fig. 8.3.

A problem which arises out of this algorithm is that an agent cannot close an environment of an annular (ring or loop) topology, of which it only sees a fragment at any given time. Closing any cell inside its field of view would violate the connectivity condition for the remaining visible environment (see Fig. 8.4a).

Solving this problem requires an algorithm for detecting and closing such loops. Considering that several agents can simultaneously attempt to close the same loop, the algorithm should include a distributed mechanism for mutual exclusion and priority resolution among agents. In

[10] In the original paper by [44], *closed* nodes are called *visited* nodes. Their naming was changed to avoid confusion.

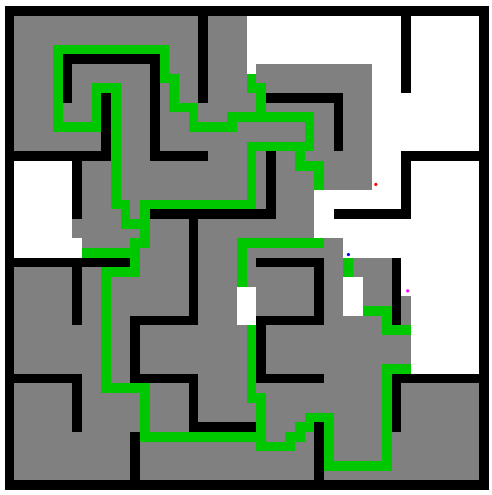
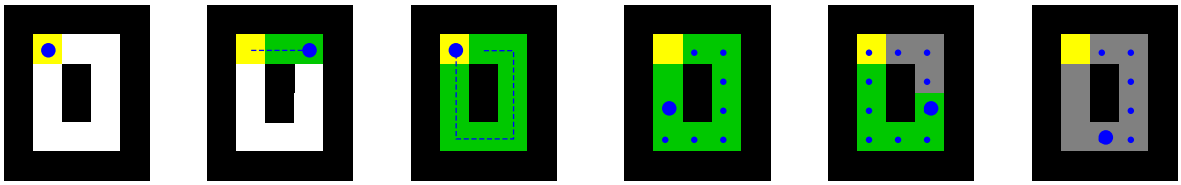


Figure 8.3 – Sample execution of the Brick&Mortar exploration algorithm. Notice how the unexplored regions are interconnected by corridors of green cells, which were left open for graph connectivity reasons.

Brick&Mortar, this translates into a 4-step algorithm for loop closing (see Fig. 8.4). Agents leave traces in the environment that enable them to detect whenever they enter a cell for a second time, which implies that they went through a loop (except for cases when they re-enter the cell from opposite direction). First, on detection of a loop, the control over it is gained by a single agent in a distributed mutual exclusion manner. Then, the agent breaks the loop by closing a part of it, while still preserving the connectivity of the environment. Finally, the agent relinquishes the loop by cleaning the marks it used for exclusive control.

Another problem with this algorithm is that the cell where agents will finish their exploration is unpredictable. The agents will not return to some exit point of the graph, which is usually expected in practical exploration missions.

The state-machine of the Brick&Mortar multi-agent exploration algorithm is presented in Fig. 8.5. The pseudocode of the algorithm is given below, in Algorithm 8.1.



(a) Exploration start (b) Ongoing exploration (c) Loop detected (d) Loop control (e) Loop closing (f) Loop cleaning

Figure 8.4 – The phases of the loop closing algorithm.

Algorithm 8.1 – Brick&Mortar exploration algorithm

```
BM_step()
{  // The agent first checks its state and
     then performs the corresponding action
   if (agentState == LOOP_DETECTION)
   {
      // Marking
      Mark the cell under the agent.

      If you have already been here (not
          coming from the opposite direction),
          and if the cell is not controlled by
          another agent, then switch directly
          to LOOP CONTROL.
      Else
         - mark the cell you are standing on.
         - update the gradient for dispersing
             the agents on this cell.

      // Navigation
      Move to open neighboring cell. Prefer
          unexplored cells over explored
          cells. Prefer those with more walls
          and closed cells around. If nowhere
          to go, close the cell you are on and
          turn OFF.
   }

   else if (agentState == LOOP_CONTROL)
   {
      Mark the cell as controlled by you,
          continuing the same path as the one
          which led you into this loop.
      When the entire loop is under your
          control, switch to LOOP CLOSING.

      If you could not control the entire
          loop, because:
         - the cell was closed by someone else,
         - you did not find your trace,
         - the loop is controled by someone
             with higher priority,
      then switch to LOOP CLEANING.

      If someone with lower priority controls
          this cell, then switch to STANDBY
          until this cell gets cleaned.
   }

   else if (agentState == LOOP_CLOSING)
   {
      Close cells of the marked loop until the
          first bifurcation.
      Then switch into LOOP CLEANING state.
   }

   else if (agentState == LOOP_CLEANING)
   {
      Clean your loop control traces, by
          moving backwards through the loop,
          while these traces exist.
      When cleaning is over, switch to LOOP
          DETECTION.
   }

   else if (agentState == STANDBY)
   {
      If the cell on which the agent stands:
         - becomes closed,
         - or is taken over by an agent with
             higher priority
      then switch to LOOP CLEANING.

      If the cell gets cleaned of other agents
          traces, then continue in LOOP
          CONTROL.

      Else remain in STANDBY.
   }

   else if (agentState == OFF)
      Agent is turned off.
} // End of "BM_step" function


/* Marks a cell while in LOOP DETECTION */
markCell(cell)
{
   If the cell is unexplored:
      - if it is blocking, then mark is as
          explored.
      - else if it is not blocking, then close
          it.
   Else if the cell was explored:
      - if it is not blocking, then close it.
}

/* Identifies if a cell is blocking inside the
    field of view of an agent*/
isBlocking(cell, field of view)
{
   A cell is blocking if:
      - if closing it would disconnect the
          open environment inside the field of
          view,
      - or if closing it would block other
          agents behind.
}
```

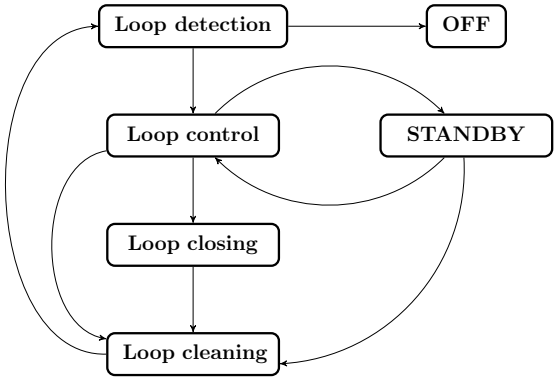


Figure 8.5 – The state machine of the Brick&Mortar algorithm.

8.4 Brick&Mortar Improved

Brick&Mortar Improved (BMI) is a new algorithm that we propose, built upon the Brick&Mortar algorithm. It introduces a new feature for gathering the agents at a rendez-vous point, once the exploration is complete. In addition, it brings some notable improvements in terms of execution time to classic Brick&Mortar. It does so by optimising the mutual exclusion algorithm performed by the agents while exploring and marking the environment. It identifies false positive occurences of loops, and avoid their closure.

8.4.1 Accelerating the mutual exclusion algorithm

The mutual exclusion algorithm that Brick&Mortar agents employ while closing loops detected in the environment has several sources of slow-down.

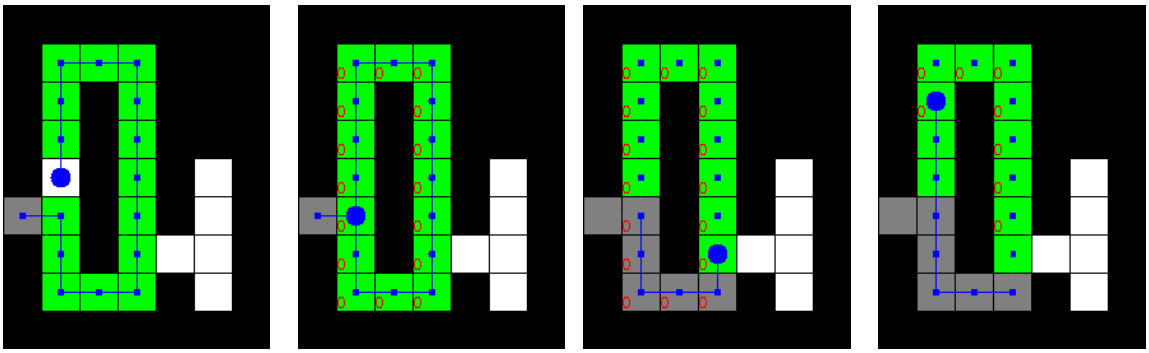
Firstly, during the *loop closing* phase, agents stop closing the cells of the loop at the first intersection with a non-closed cell, not belonging to the identified loop. This allows to maintain the connectivity of the overall graph, without risking to separate the loop from the rest of the graph. However, if an agent starts closing the loop at such an intersection, the *loop closing* phase will directly interrupt, without having closed any cell in the identified loop.

In the case of maps filled with obstacles arranged in a grid (see, for example, figure 8.12a), this poses a serious problem, as the exploration time may dramatically increase due to the frequent inefficient use of this heuristic. This is partially due to the behaviour that prioritises *loop closing* over exploration. We thus propose to improve this behaviour, when agents quit prematurely their *loop closing* algorithm.

The proposed solution is to continue the *loop closing* phase if it has started at an intersection, by skipping the intersection without closing it, and by interrupting eventually only at the second intersection (if such occurs). This solution guarantees that after each *loop closing* phase, at least one cell will be closed (at least one cell shall be added to the Tabu list). Figure 8.7 illustrates a case in which this problem arises.

Secondly, agents do not close cells behind them as they come out of a dead-end, during the *loop cleaning* phase (see figure 8.6 for an example scenario). This can be viewed as a leak in the efficiency of the algorithm. The proposed solution is simple, and implies checking whether, by closing a cell during the *loop cleaning* phase, the remaining graph stays connected inside the agent's viewing range. However, as in the case of *loop closing*, this should be done only until the first link with the rest of the graph, while moving in reverse sense, in order to maintain the graph connectivity.

(a) Brick&Mortar cell types



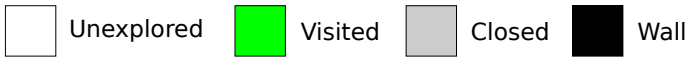
(b) An agent meets his trace and detects a loop.

(c) An agent performs the *loop control* phase, leaving his ID on the cells of the loop.

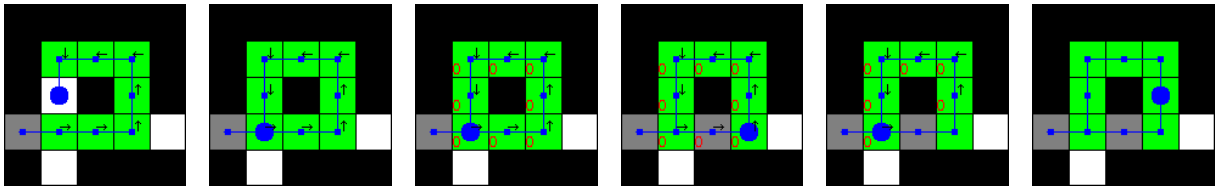
(d) An agent performs the *loop closing* phase till the first intersection with a non-closed cell neighbouring the loop.

(e) An agent performs the *loop cleaning* phase, without closing the cells that make up a dead-end.

Figure 8.6 – An example of a dead-end left open by an agent after the *loop cleaning* phase.



(a) Brick&Mortar cell types



(b) An agent on the verge of detecting the presence of a loop.

(c) An agent detects the presence of a loop.

(d) An agent ends the *loop control* phase, leaving an ID trace on the cells over which it has taken control.

(e) An agent executes the *loop closing* phase. It skips the first intersection, because the *loop closing* phase has started at an intersection, and doesn't stop until the second intersection.

(f) An agent performing *loop cleaning*, going backwards through the loop.

(g) An agent ends its *loop cleaning* phase (and quits the loop resolution algorithm), by returning to the "normal mode" of *loop detection*.

Figure 8.7 – Solution for a premature exit from the *loop closing* phase. The classic Brick&Mortar algorithm would have stopped its *loop closing* phase at step 8.7d, ending its loop resolution algorithm without having closed any of the cells in the identified loop.

Finally, agents can detect false positives (inexistent loops), when they encounter an old trail of their own. We have identified three types of loops, that agents can recognize: (1) a (true positive) loop, detected without having previously closed any cells between two visits of a cell in the loop; (2) a (false positive) loop, detected after having closed one or several cells in the *loop detection* mode (in other words, cells that weren't part of a loop); (3) a (complex true positive) loop detected having closed one or several cells in the *loop closing* phase.

Loops of the first type are normally resolved by the classic Brick&Mortar algorithm. Each of these loops contains a path, from the cell on which the loop was detected, and back to this same cell, with a length equal to the number of steps that the agent has made since it last visited it. This route is only available if the agent has not performed any cell-closing activity since the last visit of this cell. Otherwise, the loop would have been cut by these closed cells.

The second type of detectable loops, called *cut loops* (see figure 8.8), causes the Brick&Mortar algorithm to identify a false positive presence of a loop, which is in fact *cut* by the cells that were closed between the visits of the cell, that served as entrance into the loop. The agent would have consequently lost time by engaging into the costly loop resolution algorithm, which is guaranteed to stop prematurely under these conditions, without having closed any cells in the end. This implies that an agent can recognize *cut loops*, if the timestamp, that the agent left on the last cell it has closed, is more recent than the one found on the re-encountered cell.

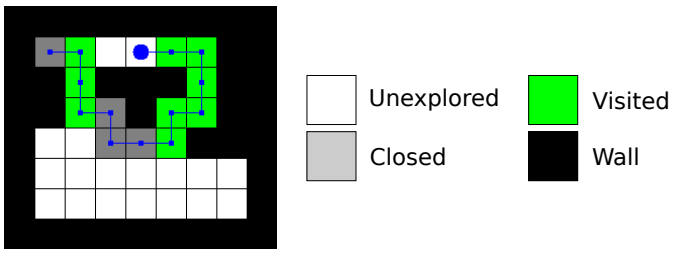


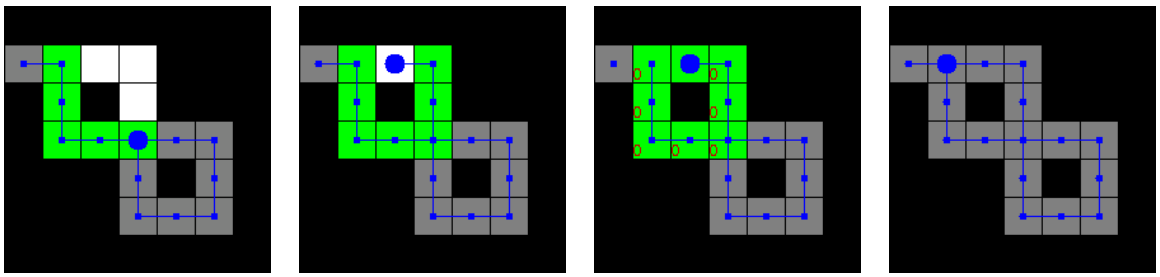
Figure 8.8 – Example of detection of a *cut loop*: an agent encounters its own trace and detects a false positive loop.

The third type of detectable loops, called *reduced loops* (see figure 8.9) contains a portion of a secondary loop that was encountered and closed during the passage of the primary loop. Such a loop is not an issue for the basic Brick&Mortar algorithm, as it doesn't cut the circuit of the loop but only reduces its length. However, this type of loops should be distinguished from *cut loops*, as they should be engaged directly once encountered, compared to the *cut loops* where loop control and closing should be avoided as being useless.

These improvements do not break the general structure of the algorithm (which was mentioned as proven to achieve complete exploration in [44]), because the state-machine used by the agents is left unchanged. Although we provide no formal proof of this statement, no deadlock was observed throughout the benchmarking phase, which consisted of over 500 runs of the algorithm.

8.4.2 Post-exploration rendez-vous

A useful property for a multi-agent ant algorithm is to be able to set a rendez-vous point on the map, where the robots should return after the completion of their exploration objective. This, for instance, can allow a team of robots exploring a building to return to the entrance, once the exploration is complete. Such a behaviour can be easily coded, so that the returning phase



(a) An agent identifies and closes a first loop.

(b) An agent identifies the second loop.

(c) An agent takes control over the second loop.

(d) The agent closes the second loop, cleans his traces and halts.

Figure 8.9 – Example of a reduced loop composed of two smaller interwoven loops.

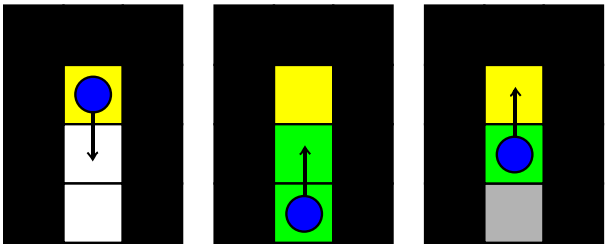


Figure 8.10 – Additional navigation and marking rules used by Brick&Mortar Improved (BMI). The attractor is treated as a non-explored region. When leaving the attractor, leave a path of explored cells back to it, as if the attractor were a non-explored region.

emerges as a result of the employed navigation and marking rules.

In the case of the Brick&Mortar algorithm, agents become aware of the completion of their exploration objective and remain motionless on their positions, because they possess no return path planification algorithm.

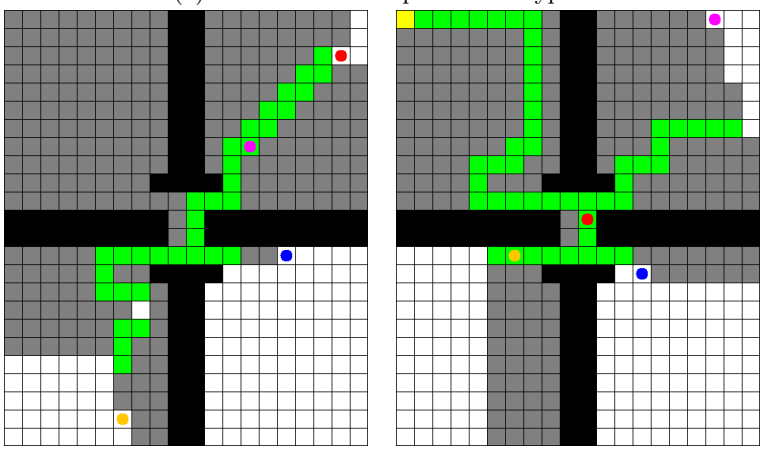
The idea is to create a new type of cell marking, called an *attractor* cell, that can be closed only when surrounded by either closed cells or walls. In the context of Brick&Mortar, this implies that the *attractor* will be the last cell to be closed. Additional navigation rules are specified, that define the attractor as an explored cell (see Fig. 8.10), as well as additional marking rules, that specify that the attractor cannot be *closed*, if it isn't surrounded exclusively by closed cells or walls (in other words, before the exploration is finished). Thus, agents will have to leave a path towards this attractor, and will have to return to it in order to complete the exploration objective.

On the implementation level, two things change in the markings employed by BMI compared to Brick&Mortar: (1) the attractor is the last cell to be closed and (2) the attractor cell is treated as an intersection during the *loop closing* phase, when occuring inside the loop resolution algorithm. A comparative example of execution of Brick&Mortar and BMI is presented in figure 8.11. The performances of BMI are analysed in more detail in section 8.4.3.

8.4.3 Experimental results of simulations in 2D environments

Our main evaluation criterion is the time it takes for a team of robots to complete the environment exploration, otherwise known as *cover time*. The cover time is influenced by the number of

(a) Brick&Mortar Improved cell types.



(b) (c)

Figure 8.11 – Comparative example of execution: Brick&Mortar vs BMI. 8.11b Execution of Brick&Mortar on the *4 rooms* map. It isn't possible to predict where will the agents be located after the exploration completion. 8.11c Execution of BMI on the *4 rooms map*. Agents leave a return path to the attractor cell.

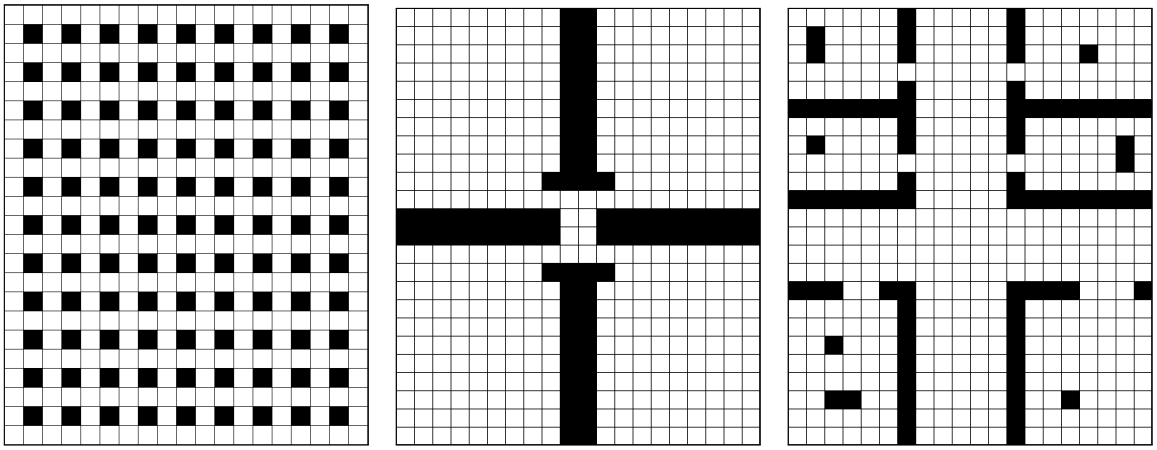
robots participating in the exploration, the size of the environment, its topology (open spaces or corridors) and by the starting point of the exploration.

For the starting position of robots, we chose to place all of them gathered at an entry point in the environment. The alternative solution of having the robots scattered at launch seems more difficult to implement in reality, as it would imply parachuting the robots over the area to explore, which is not suitable for exploring buildings and mines.

For our benchmarking results, a series of assumptions were made: (1) agents have four directions of movement (N, S, E, W); (2) agents' viewing range is 1 cell (in a 2D environment, they can see the cell underneath and the 8 neighbouring ones); (3) one cell can be shared among several agents (this was done to be able to compare our results with the ones previously presented in the literature); (4) the time consumed for leaving traces is negligible (otherwise we would have ended up counting the number of algorithm calls, instead of counting the total distance travelled by the agents); (5) the time complexity is given in the number of time-steps necessary for the agents to complete the exploration.

We have chosen 3 different types of maps, designed to evidentiate the weaknesses of algorithms: a map with obstacles of small size (figure 8.12a), a map with 4 rooms and no obstacles, each having a single entrance/exit (figure 8.12b), and an office building map (figure 8.12c). For each map type, we ran the algorithms on maps of 3 different sizes: small (1X), medium (2X), large (4X).

Three benchmarking categories have been defined, that correspond to the stages of a reconnaissance mission: (1) pure exploration algorithms, (2) algorithms capable of exploration and of identifying exploration completion, (3) algorithms capable of exploration, of identifying the exploration completion and able to gather the agents at a rendez-vous point.



(a) *Obstacles* map (b) *4 rooms* map (c) *Office* map

Figure 8.12 – Map types used for performance tests.

Environment exploration without identification of exploration termination

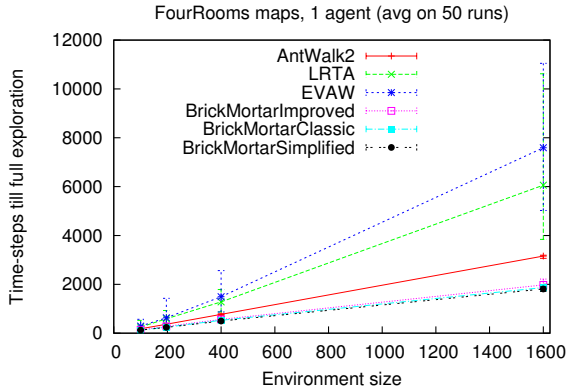
An external observer is usually used to measure the performance of exploration algorithms, as most algorithms are not capable of identifying the termination of exploration. The only exception (to our knowledge) for the multi-agent case is constituted by the Brick&Mortar algorithm. Depth-first search is also capable of identifying the termination of exploration, but only in the single agent case [180].

Given that the capacity of identifying exploration termination comes at an additional cost in terms of performance, it was decided to rewrite Brick&Mortar in such a way, so as to remove the termination identification part, leaving only the navigation and marking parts. Brick&Mortar has two components: (1) the heuristic employed for dispersing the agents: choose the non-explored cell with the largest number of surrounding walls or closed cells; apply a gradient descent method like *Node Counting* [164] to navigate among explored cells, (2) the loop resolution algorithm and the marking used for identifying the termination of exploration (the usage of a Tabu list). We created a simplified version of Brick&Mortar, that we have called Brick&Mortar Simplified (BMS), using only the environment exploration part of the algorithm, with no identification of exploration completion. This has allowed us to see if the heuristic employed by Brick&Mortar was more efficient, compared to other algorithms.

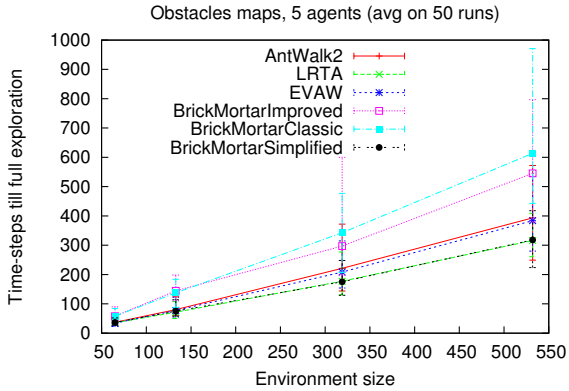
First, we have compared Brick&Mortar and BMI with the existing state-of-the-art exploration and patrolling algorithms: Ant-walk-2 (a multi-level Depth-First Search) [180], Learning Real-Time A* (LRTA*) [75, 74], and EVAW [52]. LRTA* is a gradient descent algorithm, that builds an elevation map using the pheromone traces it leaves behind, which orients the agents in their navigation. The pheromones encode the distance to the closest unexplored frontier. EVAW (a variant of the VAW algorithm [180]), also builds an elevation map, which encodes the timestamps of the last visit of each cell.

Brick&Mortar placed itself among the quickest algorithms, surpassing all the others on all maps except for the *Obstacles* map, which was specifically designed as a hard case for it (Fig. 8.13). On maps without obstacles, Brick&Mortar and its derivate algorithms behave better that all other benchmarked algorithms (Fig. 8.13a). The explication hides in the heuristic employed by Brick&Mortar for navigating among non-explored cells (i.e. choose the one with most closed cells or walls around), that allowed it to travel through the environment by sticking to its

perimeter walls, and thus generate a quasi-optimal path. The weak performance of Brick&Mortar on the *Obstacles* map is due to the prioritisation of loop resolution over exploration, and to the frequent use of the loop closing algorithm (Fig. 8.13b).



(a) Performances of analysed exploration algorithms, on maps without isolated obstacles (*Four rooms* maps used here). Brick&Mortar is by far the quickest explorer (lowest curve in this figure, almost overlayed with its variations, Brick&MortarSimplified and Brick&Mortar Improved).



(b) Brick&Mortar is the slowest algorithm on the *Obstacles* map, as it has the additional feature of detecting the exploration completion. Nevertheless, BMS (Brick&Mortar Simplified), clearly places itself among the quickest algorithms (lowest curve in this figure, overlayed with LRTA).

Figure 8.13 – Performances of analysed exploration algorithms.

Identification of exploration completion

Brick&Mortar and BMI were chosen for the performance tests in this category, as they are the only known algorithms capable of identifying the exploration completion. For a just comparison, we used a simplified version of BMI which lacked the rendezvous capacity, called Brick&Mortar Improved Simplified (BMIS).

Brick&Mortar and BMIS have the same performance on maps with no obstacles, a result that naturally confirms the theoretical expectations (as the optimised *loop closing* algorithm is never called). The results are the same, because the improvements of BMIS targeted the loop resolution algorithm, that is not emloyed on maps without obstacles.

On the other hand, the situation changes in environments heavily filled with obstacles, where the gain in time before the identification of exploration termination is clearly visible. On an *Office* map of size 20x20 (Fig. 8.12c), when varying the number of agents participating in the exploration, BMIS needs, on average, between 30% and 13% less steps to identify the termination of exploration, compared to classic Brick&Mortar (see figure 8.14).

This gain in time before the identification of exploration termination is even more evident on the *Obstacles* map (Fig. 8.12a) for the single agent case, when varying the size of the environment: the slope of Brick&Mortar is growing at almost 7 times the rate of the slope of BMI (see Fig. 8.15a). The dispersion in the exploration time of Brick&Mortar (the difference in timesteps between the slowest and quickest run) is 80x the environment size, while the dispersion of BMI is much lower, at 6x the environment size. When fixing the size of the environment at 20x20 cells, and varying the number of agents on the *Obstacles* map, BMI requires from 84% less time for the single agent case, to 50% less time for 20 agents, to identify the exploration completion

(see figure 8.15b).

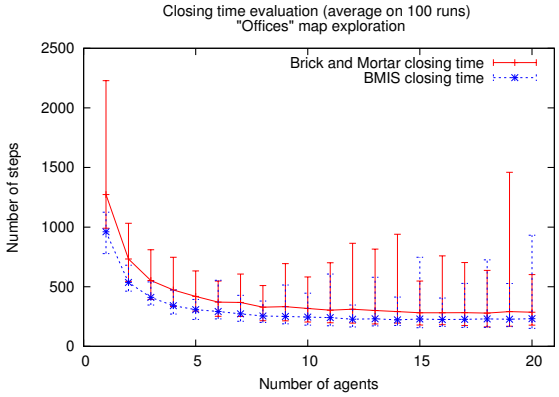
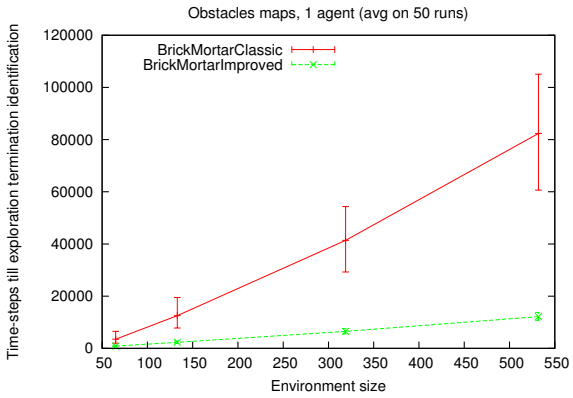
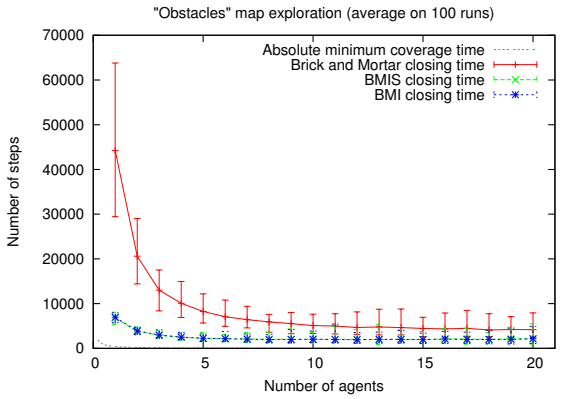


Figure 8.14 – Comparison of Brick&Mortar and BMIS performances. BMIS needs between 30% et 13% less steps to identify the termination of exploration on the *Office* map, compared to classic Brick&Mortar.



(a) The slope of BM is growing at almost 7 times the rate of the slope of BMI. The dispersion of Brick&Mortar (the difference in timesteps between the slowest and quickest run) is 80x the environment size, while the dispersion of BMI is much lower, at 6x the environment size.



(b) For the identification of exploration termination, BMIS requires from 84% less time for the single agent case, to 50% less time for 20 agents.

Figure 8.15 – Comparing BM and BMI performances on the *Obstacles* map

Post-exploration rendez-vous

This section presents the performances of BMI, that gathers the agents at the end of their exploration. Intuitively, this capacity comes at a cost, generated by the time needed to return the agents to the specified point. We should stress here that, with BMI, both identification of exploration completion and agents' rendezvous occur at the same moment in time, when all the agents gather at the attractor cell. Therefore, the additional time that BMI needs to identify the exploration completion is bounded by the length of the longest return path (equal to the number

of vertices in the diagonal of the environment), plus some variable cost induced by the time agents spend roaming through the return path, that would otherwise be closed. This overcost is clearly seen in environments without obstacles, as the *4 rooms* map (fig 8.12b). Despite this overcost, BMI does much better than Brick&Mortar on maps with lots of obstacles, due to its optimised *loop closing* algorithm. We notice that on the *Obstacles* map with size 26x26 and 532 explorable cells, in a setting with 5 agents, the time required by BMI to attain its exploration objective is on average 12% less than the time required by Brick&Mortar. (see figure 8.13b).

The overcost (in terms of time) brought by the identification of exploration completion, compared to a simple exploration of the environment, is presented in Fig. 8.16. It shows the evolution of the overcost for obstacle-filled environments of different sizes. The overcost can be calculated by subtracting the time BMS takes to simply explore the environment from the time BMIS takes to identify the exploration completion.

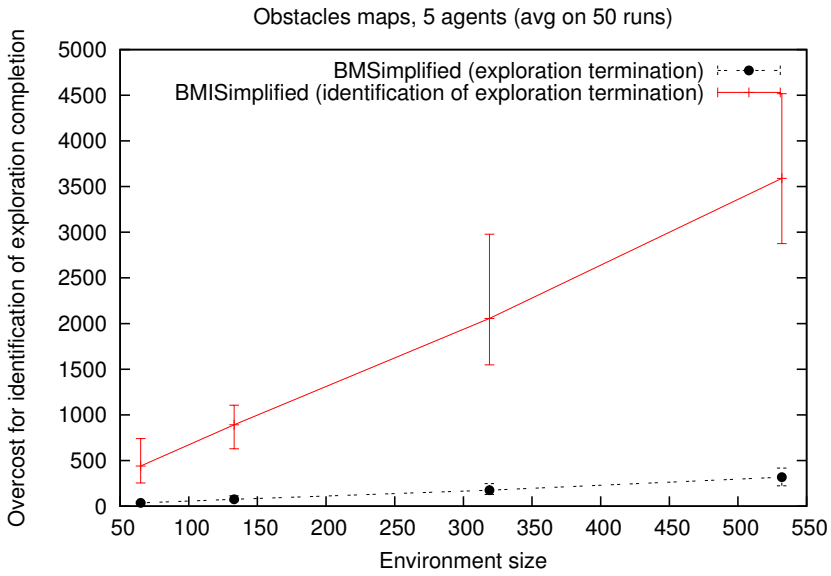


Figure 8.16 – This figure presents the overcost (in terms of time) brought by the identification of exploration completion, compared to a simple exploration of the environment. The vertical lines show the min and max time the algorithms took to complete the objective, registered over 50 runs.

8.5 Brick&Mortar Improved with Long Range Vision

The previous sections presented Brick&Mortar, a Tabu-list graph exploration algorithm, which is capable of identifying the completion of exploration. We then introduced BMI, which improved Brick&Mortar by enhancing the speed of the mutual exclusion algorithm employed by agents in their exploration and closure of the environment. It also added the capacity to gather the agents at a pre-defined point once the exploration is complete.

In this section we present BMILRV, which further upgrades this algorithm we further upgrade this algorithm, by adding navigation and marking rules for agents with long range vision capabilities. This allows agents to mark multiple cells of the environment in a single step, thus increasing the exploration speed. Another novelty is the addition of a trail to the rendez-vous point, which prevents agents from following it until the exploration is complete.

This transforms the ant-like algorithm BMI into one which uses a realistic vision representation. The visible portion of the map is calculated from each agent's perspective, as compared to a top-down view of the map in previous versions of the Brick&Mortar algorithm (see Fig. 8.19).

Agents share a common map, of which they can access only the portion surrounding them. A node can be occupied by multiple agents at the same time. Agents do not coordinate their decisions and act asynchronously. However, the marking and navigation decisions they make are considered atomic. Cell-marking cost is zero, while navigating from one cell to another neighboring cell takes 1 timestep.

In comparison to frontier-closing exploration algorithms, where agents plan their movements toward the frontiers of already explored regions, the proposed algorithm requires only minimal navigation planning. After marking the nodes in its field of view, agents move to one of the cells surrounding their current location.

The exploration algorithm discretises the surrounding world, treating it as a graph (nodes correspond to cells in a 2D grid world). This is convenient, as both 2D and 3D maps can be represented as graphs. Agents walk through this graph, trying to reduce it in size by limiting access to explored nodes. By continually decreasing the number of nodes in the open graph, while keeping it connected at the same time, agents end up gathered in the last remaining node, which is the rendezvous point (see Fig. 8.17 for an example of the algorithm execution). For clarity reasons, we shall employ 2D grids instead of abstract graphs in the rest of this section, and use the word *cell* instead of *node*.

The algorithm uses 5 types of cell markings : walls, unexplored regions, explored regions, closed regions, and the rendezvous point (see Fig. 8.18). Unexplored and explored cells form the open parth of the map, in which agents are authorized to navigate. The closed cells are the ones added to the taboo-list, and to which access is forbidden.

At each time step, each agent performs 2 activities: marking and navigation. During marking, the agent considers all the cells inside its field of view (Fig. 8.19). If the removal of a cell doesn't break the connectivity of the remaining open portion of the map, then this cell is declared closed for further access. If a cell is necessary for maintaning the connectivity of unexplored regions, this cell is declared as *explored* and left open for further access. Thus, a cell is left open if:

- it is at the edge of the field of view, potentially linking the environment outside the field of view to the one inside it;
- it is adjacent to any non-visible cells inside its field of view (i.e. shadows) which potentially hide unexplored cells;
- it disconnects the map open for access inside the agent's field of view.

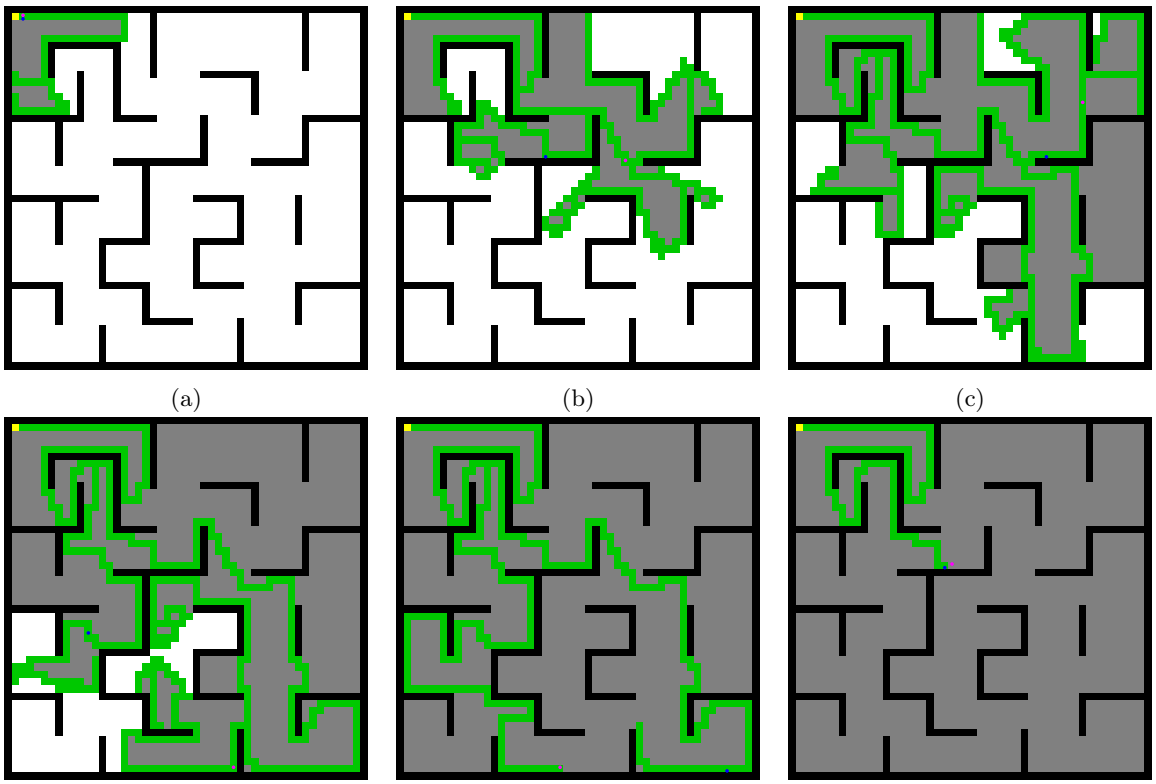


Figure 8.17 – Exploration performed by two BMILRV agents. The rendezvous point is in the top-left corner of the map.

Figure 8.18 – Cell types used by BMILRV

Considering the multi-agent nature of the algorithm, a 4th condition is required: a cell cannot be closed if that will block other agents. This includes checking for the presence of robots or their mutex traces in the cells surrounding the cell being analyzed.

The order in which the cells inside an agent's field of view are closed influences the shape of paths that remain open for continuing the exploration of the graph. In Fig. 8.19, for instance, the cells were analyzed in clockwise direction, from the ones closest to the agent to the farthest ones, starting from the 12 o'clock position.

After the agent has marked the cells inside its viewing range, it will mark the cell underneath itself. If this cell is left open, the marking will also contain the direction followed to exit this cell. The agent will also set the value of the dispersion gradient inside this cell, which is used to direct agents between explored cells left open for navigation. Initially, the dispersion gradient value for all the cells is set to 0. Updating the dispersion gradient value of a cell assigns it the minimal surrounding gradient value plus one, a policy known as LRTA* in the literature [74]. This pushes agents down the dispersion gradient to the surrounding cells.

It may happen that several agents share the same cell (particularly in the beginning of exploration), when only the first agent that marks the surrounding environment gets to influence it. If a single exit path is left open by the first agent, the remaining agents will have to follow it,

preventing their dispersion. This behavior can be avoided by forcing agents to limit their marking range to the distance between them and their closest neighbor inside their field of view. If no other agent falls inside their field of view, then the normal marking range is used. Experimental results have shown that this decreases the time till full exploration of the environment, as determined by an external observer.

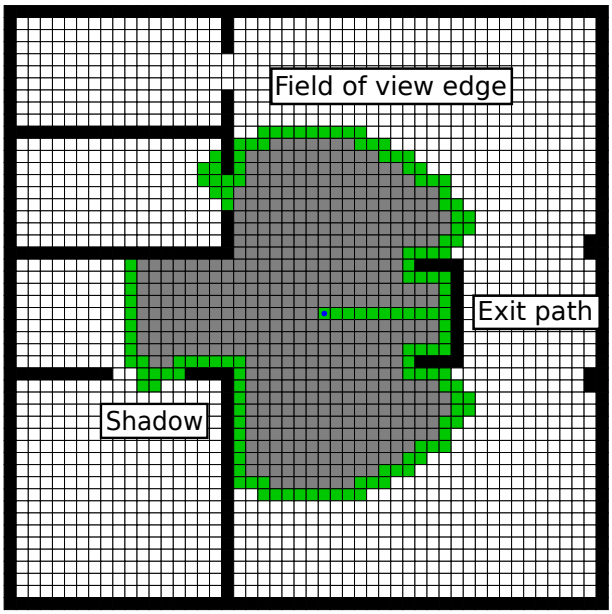


Figure 8.19 – BMILRV marking illustration. All the cells in the viewing range are closed, except those on its edge, and those near the shadows of objects. An exit trail is implicitely left by the algorithm, as its absence would have cut the agent from the rest of the environment. Its shape depends on the order in which the cells in the viewing range are closed. All the unexplored regions remain accessible.

In the navigation step, the agent moves to one of the open neighboring cells, giving priority to unexplored cells over the explored ones. If it travels from an explored cell to another explored cell, it will follow the dispersion gradient, which will dictate its direction of movement.

We also improved the exploration by preventing the agents from going to the rendezvous point if the exploration hasn't yet completed. This is done when there is a unique way that leads from the rendezvous point to the rest of the graph. It was implemented using the dispersion gradient that agents employ to disperse themselves in the environment. Maximum gradient values are set to the cells composing this unique path leading to the rendezvous point, which prevents access to them for the agents performing gradient descent, unless they have no other choice.

The pseudocode of the BMILRV multi-agent exploration algorithm is given below, in Algorithm 8.2. The state-machine describing this algorithm is the same as the one of Brick&Mortar, presented in Fig. 8.5 on page 107.

Algorithm 8.2 – Brick&Mortar Improved Long Range Vision exploration algorithm

```
BMILRV_step()
{  // The agent first checks its state and
     then performs the corresponding action
   if (agentState == LOOP_DETECTION)
   {
      // Marking
      Repeat
      {
         Mark all open cells inside your field
             of view (except the one you are
             standing on, to avoid being
             blocked).
      } while at least one cell was closed
          inside field of view

      If you have already been here (not
          coming from the opposite direction),
          and if the cell is not controlled by
          another agent, and if you have not
          closed cells since your last visit
          to this cell, then switch directly
          to LOOP CONTROL.
      Else
         - mark the cell you are standing on.
         - update the gradient for dispersing
             the agents on this cell.

      // Navigation
      Move to open neighboring cell. Prefer
          unexplored cells over explored
          cells. Prefer those with more walls
          and closed cells around. If nowhere
          to go, close the cell you are on and
          turn OFF.
   }

   else if (agentState == LOOP_CONTROL)
   {
      Mark the cell as controlled by you,
         continuing the same path as the one
             which led you into this loop.
      When the entire loop is under your
          control, switch to LOOP CLOSING.

      If you could not control the entire
          loop, because:
         - the cell was closed by someone else,
         - you did not find your trace,
         - the loop is controled by someone
             with higher priority,
      then switch to LOOP CLEANING.

      If someone with lower priority controls
          this cell, then switch to STANDBY
          until this cell gets cleaned.
   }

   else if (agentState == LOOP_CLOSING)
   {
      Close cells of the marked loop until the
          first bifurcation after the place
          where you started the closing phase.
      Then switch into LOOP CLEANING state.
   }

   else if (agentState == LOOP_CLEANING)
   {
      Clean your loop control traces, by
          moving backwards through the loop,
          while these traces exist.
      When cleaning is over, switch to LOOP
          DETECTION.
   }

   else if (agentState == STANDBY)
   {
      If the cell on which the agent stands:
         - becomes closed,
         - or is taken over by an agent with
             higher priority
      then switch to LOOP CLEANING.

      If the cell gets cleaned of other agents
          traces, then continue in LOOP
          CONTROL.

      Else remain in STANDBY.
   }

   else if (agentState == OFF)
      Agent is turned off.
} // End of "BMILRV_step" function


/* Marks a cell while in LOOP DETECTION */
markCell(cell)
{
   If the cell is the rendezvous point, then
       leave no mark on it.
   Else if the cell is unexplored:
      - if it is blocking, then mark is as
          explored.
      - else if it is not blocking,
          then close it, and update the time of
              the last cell closing.
   Else if the cell was explored:
      - if it is not blocking, then close it,
          and update the time of the last cell
          closing.
}
```

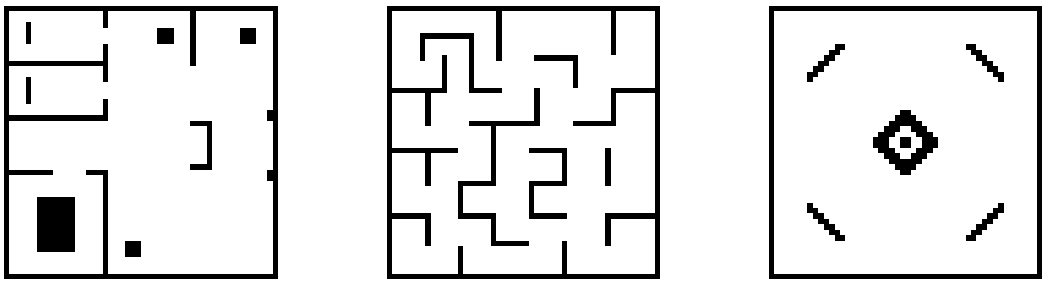
```
/* Identifies if a cell is blocking inside the
    field of view of an agent*/
isBlocking(cell, field of view)
{
   A cell is blocking if:
      - it is at the edge of the field of view,
      - or it is adjacent to a cell in shadow
          (non-visible cell behind an
          obstacle),
      - or if closing it would disconnect the
          open environment inside the field of
          view,
      - or if closing it would block other
          agents behind.
}
```

8.5.1 Experiments with agents with long-range vision

We evaluated the performance of BMILRV in a series of simulations on maps mimicking human environments, such as offices cluttered with obstacles, mazes and gardens (see Fig. 8.20). BMILRV is currently the only known multi-agent algorithm where agents use a shared map, have a local vision of this map, don't communicate their exploration targets, and, most importantly, identify the end of exploration. No comparable algorithm of the same class exists, to our knowledge.

However, in the absence of comparable state-of-the-art algorithms, we can still evaluate BMILRV relative to some baseline. In this sense, we used a group greedy frontier explorationg algorithm as baseline. This is a sub-optimal heuristic, which avoids the complexity of solving a Hungarian algorithm to calculate an optimal assignment of agents to frontiers. It assigns agents to frontiers in a greedy manner, incrementally selecting for assignment pairs of agents and frontiers with minimal distance between them.

In both cases agents communicate to share a common map (in BMILRV, this is done via virtual pheromone traces). However, in comparison to BMILRV, frontier exploration algorithms make 2 additional strong assumptions. First, agents communicate between themselves, so as to distribute the exploration effort by selecting different frontiers to explore, and thus avoid following each other. Second, agents use their knowledge of the entire map (as opposed to knowledge of only the visible surroundings in BMILRV) to plan optimal navigation routes.



(a) Office (b) Maze (c) Garden

Figure 8.20 – Maps explored during the experiments.

The parameters that varied in our experiments were: the number of agents performing the exploration, and the size of the map. The viewing range of agents in all our experiments was fixed at 10 grid cells. The results are shown in Fig. 8.21.

Increasing the number of exploring agents leads to shorter exploration times, as seen in Fig. 8.21a. This means that the algorithm is able to distribute the exploration effort among its agents. However, exploration efficiency is also dependent on the size and topology of the environment, which can get saturated with agents. In Fig. 8.21a, the environment gets saturated at 5 agents,

after which adding new agents doesn't decrease the exploration time. As compared to the greedy Frontier Exploration algorithm, BMILRV is slowed down by obstacles present in the environment, which force agents to go through the costly loop closing algorithm, in order to close the loops that form around such obstacles. Navigation in the environment is also suboptimal for BMILRV, as the paths left open by agents are not of shortest possible length.

Fig. 8.21b shows the number of steps required for exploration completion and rendezvous in maze-type environments. Again, BMILRV is slowed down by the number and size of isolated obstacles present in the environment. Better comparative results are obtained for environments of type *Garden*, where the number and size of obstacles is reduced, as compared to *Maze* maps (see Fig. 8.21c).

8.6 Future work on distributed exploration algorithms

All the presented experiments were performed in simulation. Future work will therefore include the implementation of the new algorithm on a robotic platform, that would communicate with a grid-type environment, such as a network of intelligent tiles like the ones presented in chapter 3, or any other type of sensor network. This will raise the issue of uncertainty in sensing, localisation, and in the movement of robots.

Regarding the uncertainty in localisation, considering that BMILRV agents do not maintain a global map in their memory, the environment itself could help agents localise themselves (as presented in chapter 7). By tracking the robots during their exploration, the floor can uniquely identify them, and communicate them their exact location.

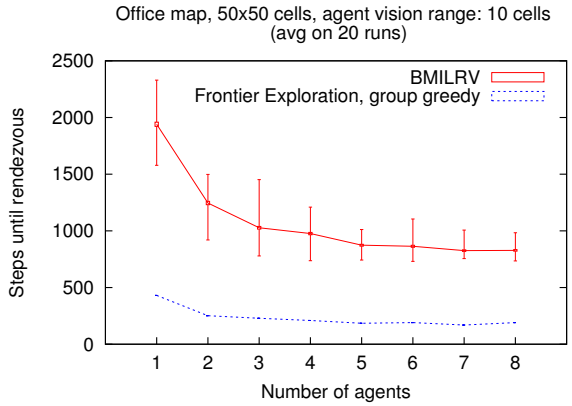
The uncertainty in robot movements impacts their location. Errors in localisation generated by erroneous movements can be identified by the mismatch between the expected location of the robot and the one communicated by the environment. Robots could then attempt to return to their intented location. Another solution would be to allow the robot to reset the markings in the visible portion of the environment surrounding the robot, allowing it to move out of any region it might have stepped into, including previously closed regions of the environment.

The sensing uncertainty would influence the way the cells of the environment are perceived and marked. To prevent errors, the graph connectivity conditions could be checked by the intelligent environment itself, which is supposed to have a perfect knowledge of its topology and of all the cell markings.

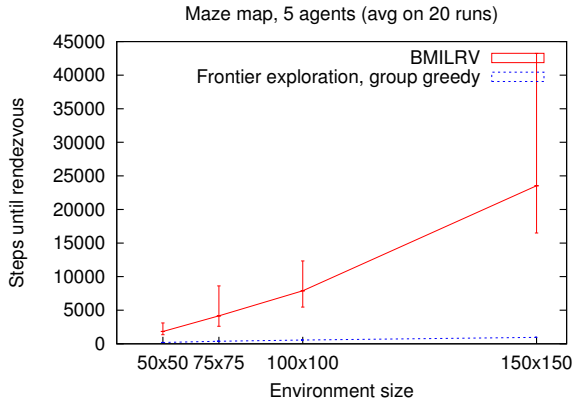
The long range vision capability of BMILRV brings with it an acceleration in marking speed. Compared to Brick&Mortar Improved, where agents only mark the cell underneath themselves, in BMILRV the marking process is accelerated by a factor proportional to (half) the perimeter of the field of view.

However, the long range vision does not influence the way agents close loops around obstacles, since obstacles will always occlude a portion of the environment behind them, as seen from agents' perspective. Nevertheless, the situation is different for agents with a top-down viewing perspective (e.g. aerial vehicles), from which obstacles generate no occlusions or shadows. In this case, obstacles of size inferior to the field of view will not generate any loops at all.

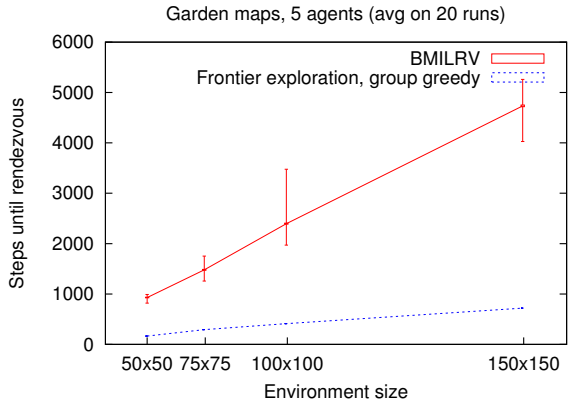
The virtual marking system has the potential to indirectly solve navigation problems. Local marking and navigation rules are sufficient to navigate through the environment and allow the agents to avoid costly path calculations to the closest non-explored regions. Future work will also focus on a better dispersion of exploring agents. Agents could reshape the corridors left open to keep the non-explored regions connected, and which are often sub-optimal in length. The impossibility of deadlock among agents is also a property we would like to prove.



(a) Time-steps till exploration completion and rendezvous on the office map (Fig. 8.20a). Size of the map: 50x50 pixels. Agent viewing range: 10 cells. BMILRV is capable to distribute the exploration effort.



(b) Time-steps till exploration completion and rendezvous on Maze maps (Fig. 8.20b). Exploring agents: 5. Agent viewing range: 10 cells. The size and number of isolated obstacles present in the environment heavily impact the efficiency of BMILRV.



(c) Time-steps till exploration completion and rendezvous on Garden maps (Fig. 8.20c). Exploring agents: 5. Agent viewing range: 10 cells.

Figure 8.21 – Experimental results obtained using simulated map explorations. The performances of BMILRV and greedy Frontier exploration are shown together for illustration purposes only. These algorithms are of different categories and are not directly comparable.

8.7 Conclusion

We have presented a multi-agent algorithm for exploration of unknown environments, based on a tabu-list approach. Agents mark explored parts of the environment as closed for access, while keeping the unexplored regions interconnected until the end of exploration. As agents can only travel through open paths in the environment, they decide on the direction of the next movement without using high-level planning.

Our contributions are manifold: we generalized the algorithm to agents with variable viewing ranges, allowing them to mark multiple cells at a time; we added the possibility to gather the agents once the exploration is complete, by defining a rendezvous point; we increased the exploration speed by preventing agents from going to the rendezvous point before exploration is complete, by improving the mutual exclusion algorithm agents use when closing cells in the the environment, and by detecting false positive occurences of loops.

With the proliferation of environments with embedded memory capacities, integrated in ambient intelligence systems, stigmergic algorithms regain their practicality for robotic applications. The guidance of robots by the environment (the floor in our case), both in providing occupancy and Voronoi maps for navigation, and in their exploration of the environment, can simplify the programming and deployment of robots. This is particularly true for blind or short-sighted robots, like autonomous vacuum cleaners, which currently use only close-range sensors, and could benefit from the mapping and memory service offered by the environment.

This chapter concludes the III part of the thesis, in which we have presented proofs of concept for the integration of a sensing floor with the robotic actuators of a habitat. In Chapter 7, we showed how an omnipresent load sensor can generate occupancy maps of the environment, and how it can compute the safest navigation paths based on these maps. In Chapter 8, we exploited the memory embedded in the environment to support stigmergic algorithms. We used the example of stigmergic distributed exploration algorithms, where agents leave traces in the environment to perform a tabu-list exploration. It is now time to move on to future perspectives on the usage of sensing floors in ambient intelligence.

Finale

9

Conclusion and future work

Contents

This thesis explored the capabilities of pressure-sensing floors as a tool for ambient intelligence. We gave an extensive account of the state-of-the-art in sensing floors, offering design suggestions stemming from the experience we gained during these 3 years of research.

In ambient intelligence settings, where human inhabitants populate the environment, scene interpretation is required before any autonomous action is performed. We therefore developed and introduced techniques for tracking and recognising people and objects in the environment, using the weight information provided by load-sensing floors. The problems that emerged on modular, tiled floors, like the interpretation of spread loads, when objects span several tiles on a modular floor and have multiple points of support, have been solved using a new object segmentation algorithm proposed in this thesis.

As it may be expected, ambiguities may arise between objects of equal weight, preventing their correct localisation when using only their weight. We therefore presented an innovative technique for scanning the surfaces of objects in contact with the ground, so as to use them for object recognition. For this purpose, we have drawn analogies between pressure sensing and light sensing, which opened the door to the use of computer vision techniques to perform pressure sensing. However, further research is needed to loosen the algorithm's constraints on the movement of objects, so as to be able to use it in real life situations.

Ambient intelligence systems have, besides sensors, robotic actuators that can interact with the supervised environment. The navigation and obstacle avoidance algorithms of these mobile robotic actuators are usually based on data from high-precision sensors like laser range-finders, which also happen to be expensive. In this context, one way of representing the safest navigation routes in terms of obstacle avoidance is by constructing a Voronoi diagram of the environment. We have shown that sensing floors can generate occupancy maps of the perceived environment,

and compute Voronoi diagrams, that are then transmitted to mobile robots for navigation planning. We have employed a distributed, asynchronous computation method that can still calculate a Voronoi diagram despite the desynchronisation between the tiles of the floor. Externalising this mapping function into the environment allows the use of mobile robots with fewer sensors and a simpler design.

Exploration of environments is another task routinely performed by teams of robots. We have shown that by embedding memory capacities into the environment it is possible to support multi-agent algorithms based on stigmergy, where agents leaves traces in the environment. These traces help agents in their decision making, as in the example of navigation and exploration algorithms. We improved an existing algorithm capable of distributively identifying the end of exploration by adding the support for post-exploration rendez-vous points, and optimised the mutual exclusion algorithm employed by the agents in their exploration of the environment. We finally generalised the approach to agents with an arbitrary viewing and marking range, that perceive the environment from an agent-based perspective.

We also encountered obstacles while conducting this work. In the absence of an adequately placed acceleration sensor, we could not directly distinguish the pressure force due to mass from the pressure force due to acceleration. A consequence of this was the need to track objects in order to calculate their mass and recognise them. It also was not possible to distinguish objects of equal mass without adding new discriminative features like their surface, as was suggested in Chapter 5.

The high sensor noise present in our prototype did not allow us to detect lightweight objects under 2 kg of mass. This pointed to the necessity of using probabilistic filtering when reading the pressure values for static objects. However, existing filtering techniques weren't of much use during dynamic activities performed by humans, as they require models of the activities to generate a filtered pressure value. Performing a parallel analysis of the pressure readings using all the existing activity models may provide hints to the most appropriate model to use, although this is computationally expensive. A co-authored paper on probabilistic localisation techniques has been submitted to ICRA 2016 [126].

Interfacing our sensing floor with ROS gave us the possibility to easily interconnect it with the other sensors and robots in the environment. In particular, we used it to provide navigation guidance to robots in simulation.

Regarding the design of the sensing floor, the concerns that need to be taken into consideration when creating a prototype from scratch are presented in the following section.

9.1 Remarks on the design of sensing floors

The experience gathered after 3 years of work with our first prototype has led us to several conclusions regarding the design of pressure-sensing floors. We analysed their deployment, their shape, and functionalities.

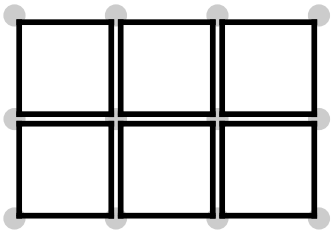
Considering the installation of a sensing floor, we identified two categories of sensors: modular floors (i.e. tiled floors), and monolithic sensors (e.g. carpets [137]). The installation of monolithic sensors is quick, as they have to be simply unrolled onto the floor and plugged. In comparison, the installation of modular floors is more complex, as each component needs to be installed and connected separately. It is practical to have modules with integrated cabling, which can also transmit electricity and network connection to the neighboring modules (as seen in Z-tiles [123]). This avoids having separate electricity and network connectors and plugs for each tile or floor

module.

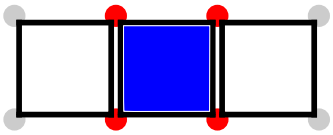
In terms of maintenance, modular floors present a series of advantages: hardware errors can be easily localised, and modules can be easily transported and replaced. From a practical standpoint, the design should be water-proof, to allow wet washing and cleaning. Floors could also be energy-harvesting, reducing the need for an energy source [35].

Concerning sensing capabilities, the smaller the floor tiles are, the higher is the sensing resolution, and the better is the segmentation of objects on the floor. The use of the tiles dictates their smallest practical size: for human foostep tracking applications, the adequate size of a tile would be the size of a foot (approx. 30 cm × 30 cm). For more fine-grained details, as required by biometrical applications, other types of sensing floors may be more adequate (e.g. pressure mats), if judged by price per unit of sensing surface, or by their fabrication complexity. It would also be interesting to have sensors that capture the xyz components of the ground force. This would allow the reconstruction of the human body posture, given a model of the human body and of its constraints [190, 118].

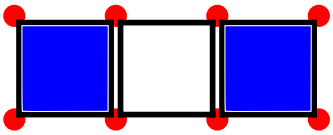
An intuitive solution for reducing the number of sensors employed in the floor is to implement sensor sharing between tiles, as shown in Fig. 9.1a. When the density of objects on the floor is low, leaving free space around objects, the reduced number of sensors suffices for object segmentation. In this case, the free tiles that neighbour the occupied tile will have only a part of their load sensors activated, allowing to dismiss them as non-occupied tiles (see Fig. 9.1b). However, when the density of objects on the floor is high, leaving little free space between objects, a floor with sensor sharing will not be able to segment objects as well as a floor without sensor sharing. In this case, free tiles between occupied tiles will have all their load sensors activated, falsely identifying themselves as occupied (see Fig. 9.1c).



(a) A typical way for implementing sensor sharing between tiles. The sensors (shown as red circles) are shared between 4 square tiles that have neighbouring corners.



(b) Free tiles (in white) neighbouring an occupied tile (in blue) will have only half of their load sensors activated (red), allowing to identify them as non-occupied tiles.



(c) A free tile (in white) between occupied tiles (in blue) will have all its load sensors activated (red), falsely identifying itself as occupied.

Figure 9.1 – Sensor sharing between tiles as a solution for reducing the number of required sensors.

In our sensing floor prototype, the tiles supporting heavy furniture like wardrobes and kitchen cabinets were not equipped with sensors, creating additional regions where objects may disappear from the floor's view. Although it would be possible to track objects by their last observed position, it would have been better to have an omnipresent sensing floor, covering the entire surface of the house. This would leave a minimal amount of entry/exit points where objects may disappear, which would correspond only to doors and windows.

Another important aspect is the simulation of the sensor. The problem here is to calculate the distribution of a force onto the sensors supporting the tile. Isostatic rigid tiles have the advantage of requiring only static equilibrium equations for simulating the distribution of forces

on their supports. Tiles with more points of support are hyperstatic, and require the application of computationally expensive numerical techniques (e.g. finite element method) to calculate approximate solutions for the distribution of forces on the sensors. On the other hand, the more sensors are involved in the calculation of the center of pressure (COP), the more the probability distribution of measurement noise resembles a Gaussian, due to the Central Limit Theorem.

Regarding the sensor sampling frequency, for people walking or running on a tiled sensing floor, most of the load signal energy lies below 250 Hz [3]. This establishes a practical upped bound for the sampling frequency of sensing data.

In the next and last section of this thesis, we will present perspectives for future research.

9.2 Future work

Several directions are envisioned for future work with sensing floor prototypes, orbiting around multi-modal data processing, activity recognition, biomedical analysis, semantic mapping of the environment, and interaction with robots. These prospective topics of research are detailed below.

9.2.1 Extraction of gait parameters for medical analysis

For medical applications, the floor could provide high-precision extraction of footsteps, and of derivate data about the angle between the feet and the direction of movement, pressure distribution between the two feet, and so on. These parameters serve as medical indicators of the state of human gait. Calculating statistics on the human activity (number of steps, walking speed, number and type of interactions with objects, weight lifted per day, etc.) over a fixed time span (a day, a month, a year) would also present a rich source of information.

9.2.2 Object recognition using sensing floors

It would be interesting to continue exploring if objects on the floor can be recognized by the shape of their contact surface, in minimally constrained movement conditions. Is it possible to identify the position and orientation of an object on the floor, given a model of its surface and of the distribution of weight inside it ?

Automatic measurement and introduction of new persons and objects to the list of known entities would help bootstrap newly installed sensing floors with data about the initial occupants, and would keep it working with the addition of new users. Measuring the presence frequency of each user would allow to dynamically estimate the probability of detecting a given user in the environment, and reduce the computational load on the user localisation algorithm.

When performing object tracking, the floor could separate objects into layers. Each newly detected object would be assigned to a new layer. This layered view of the objects exerting pressure on the floor would improve their segmentation in proximity conditions, as it happens when users walk near objects, or interact with them.

9.2.3 Multi-modal data processing

Combined with depth cameras, sensing floors can non-intrusively recognize a wide palette of activities and human states. This would meet the practical requirements of hospitals, such as the identification of falls, of people lying on the ground, as well as tracking the postures of patients to evaluate their daily activity pattern.

Sensing floors can help cameras to track their targets, whenever these get out of view. This happens particularly often in environments with furniture, that visually oclude objects either partially or entirely. In this way, persons re-entering the cameras' field of view can be re-identified without using complex recognition algorithms.

Pressure sensing floors can also improve multi-modal activity recognition, adding discriminating information about the weight supported by the limbs. Triaxial accelerometers embedded in the floor tiles may add a new dimension for differentiating between the dynamic activities performed by humans.

Although there are existing datasets for recognizing activities of daily living [129], there is currently no standard dataset for benchmarking multi-modal activity recognition software, which would involve the force sensing aspect. Creating such a dataset, using data from depth cameras, load sensors on the floor, and motion tracking cameras, is one of our future goals in our quest for better multi-modal activity recognition. We also plan to enrich the dataset with data on extracted human skeletons and the position of limbs.

9.2.4 Activity recognition

The recognition of cases of emergency may require the deployment of specific sensors for evaluating the health-state of a human in need. Mobile robots can help deliver the required sensing capabilities to where these are needed, as in the case of people lying on the ground (see Fig. 9.2).

Figure 9.2 – Investigating the health state of a person using a mobile robotic actuator, after a fall detected by the sensing floor.

We would like to exploit activity grammars in our automatic recognition of activities. These can compress the recorded signals, automatically finding the redundant gestures composing an activity [139]. Such an approach can potentially generate representations of activities that can be easily interpreted by a human expert.

9.2.5 Semantic mapping and interaction with robots

One of the next steps in developing our ambient assisted living project would be the addition of semantic mapping capabilities. Ontologies could allow to infer the functional usage of a room by the type of activities that happen inside (e.g., balcony used for eating, room used for sleeping), or by the set of objects that it contains (e.g., dishes and cooking instruments are stored in the kitchen) [112].

Semantic mapping may also be useful for the implementation of stigmergic exploration algorithms, using mobile robots deployed on the SmartTiles sensing floor. Embarking the developed algorithms on board the mobile assistant robots would confront the algorithms with real-life conditions, forcing us to make further improvements towards their practical use.

9.2.6 Domains of application

Ambient intelligence can be applied to domains where it is important to maintain some specific conditions, and to monitor the activities inside a given space (e.g. hospitals, retirement homes). In kindergartens, for instance, this would allow to survey that no child is involved in dangerous activities. It could also be used for surveillance inside buildings and warehouses, for detecting suspicious or criminal activity. Management and surveillance of an environment could one day be required on board large transport vessels. In a wider view, ambient intelligence could be used city-wide, providing technical instruments for automatically tracking down criminals.

To conclude, we think that sensing floors have a promising future in ambient intelligence, as floors are one of the most convenient placements for omnipresent pressure sensors. Non-intrusive but informative enough to perform basic surveillance tasks, they are the adequate sensor for spaces where privacy is a requirement. Speaking about ambient intelligence, which is still in its infancy years, a holistic view of an ambient intelligence surveilling and acting in a bounded space is necessary to understand, research and develop the capabilities of an omnipresent, omniscient system. Living in such an environment can be seen as living inside a robot that has control over the space that we inhabit.

Bibliography

[1] Andrew Adamatzky and Benjamin de Lacy Costello. "Collision-free path planning in the Belousov-Zhabotinsky medium assisted by a cellular automaton". In: *Naturwissenschaften* 89.10 (2002), pp. 474–478.

[2] Andrew Adamatzky and Benjamin de Lacy Costello. "Reaction-diffusion path planning in a hybrid chemical and cellular-automaton processor". In: *Chaos, Solitons & Fractals* 16.5 (2003), pp. 727–736.

[3] M.D. Addlesee, A. Jones, F. Livesey and F. Samaria. "The ORL active floor". In: *Personal Communications, IEEE* 4.5 (1997), pp. 35 –41. ISSN: 1070-9916. DOI: 10.1109/98.626980.

[4] I. Al-Naimi, Chi Biu Wong, P. Moore and Xi Chen. "Advanced approach for indoor identification and tracking using smart floor and pyroelectric infrared sensors". In: *Information and Communication Systems (ICICS), 2014 5th International Conference on*. 2014, pp. 1–6. DOI: 10.1109/IACS.2014.6841966.

[5] G. Alankus, N. Atay, Chenyang Lu and O.B. Bayazit. "Adaptive Embedded Roadmaps For Sensor Networks". In: *Robotics and Automation, 2007 IEEE International Conference on*. 2007, pp. 3645–3652. DOI: 10.1109/ROBOT.2007.364037.

[6] Mihai Andries and François Charpillet. "Multi-robot exploration of unknown environments with identification of exploration completion and post-exploration rendezvous using ant algorithms". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ. Tokyo, Japan, 2013.

[7] Mihai Andries and François Charpillet. "Multi-robot taboo-list exploration of unknown structured environments". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*. Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)September 28 - Oct 2, 2015 (Hamburg, Germany). Hamburg, Germany, 2015. URL: https://hal.inria.fr/hal-01196008.

[8] Mihai Andries, François Charpillet and Olivier Simonin. "High resolution pressure sensing using sub-pixel shifts on low resolution load-sensing tiles". In: *IEEE International Conference on Robotics and Automation (ICRA), 2015*. 2015.

[9] Mihai Andries, François Charpillet and Olivier Simonin. "Localisation of humans, objects and robots interacting on load-sensing floors". In: *Sensors Journal, IEEE* PP.99 (2015), pp. 1–1. ISSN: 1530-437X. DOI: 10.1109/JSEN.2015.2493122.

[10] Mihai Andries, Nassim Kaldé, Olivier Simonin and François Charpillet. "Roadmap extraction service provided by a sensing floor for robotic navigation". (work in progess). 2015.

[11] Jan Anlauff, Tobias Grosshauser and Thomas Hermann. "tacTiles: A Low-cost Modular Tactile Sensing System for Floor Interactions". In: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. NordiCHI '10. Reykjavik, Iceland: ACM, 2010, pp. 591–594. ISBN: 978-1-60558-934-3. DOI: 10.1145/1868914.1868981. URL: http://doi.acm.org/10.1145/1868914.1868981.

[12] M.H. Baeg, Jae-Han Park, Jaehan Koh, Kyung-Wook Park and Moon-Hong Baeg. "RoboMaidHome: A Sensor Network-based Smart Home Environment for Service Robots". In: *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*. 2007, pp. 182–187. DOI: 10.1109/ROMAN.2007.4415077.

[13] Laurent Ballaz, Maxime Raison, Christine Detrembleur *et al.*. "Decomposition of the vertical ground reaction forces during gait on a single force plate". In: *Journal of musculoskeletal & neuronal interactions* 13.2 (2013), pp. 236–243.

[14] N.M. Barnes, N.H. Edwards, D.A.D. Rose and P. Garner. "Lifestyle monitoring-technology for supported independence". In: *Computing Control Engineering Journal* 9.4 (1998), pp. 169–174. ISSN: 0956-3385. DOI: 10.1049/cce:19980404.

[15] M.A Batalin, G. Sukhatme and M. Hattig. "Mobile robot navigation using a sensor network". In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. Vol. 1. 2004, 636–641 Vol.1. DOI: 10.1109/ROBOT.2004.1307220.

[16] Antoine Bautin, Olivier Simonin and François Charpillet. "MinPos : A Novel Frontier Allocation Algorithm for Multi-robot Exploration". In: *Intelligent Robotics and Applications*. Vol. 7507. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 496–508. ISBN: 978-3-642-33514-3. DOI: 10.1007/978-3-642-33515-0_49. URL: http://dx.doi.org/10.1007/978-3-642-33515-0_49.

[17] Antoine Bautin, Olivier Simonin and François Charpillet. "SyWaP: Synchronized Wavefront Propagation for multi-robot assignment of spatially-situated tasks". In: *ICAR 2013 : International Conference on Advanced Robotics*. Uruguay, Nov. 2013, p. 7.

[18] Z.T. Beattie, Chad C. Hagen, M. Pavel and T.L. Hayes. "Classification of breathing events using load cells under the bed". In: *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. 2009, pp. 3921–3924. DOI: 10.1109/IEMBS.2009.5333548.

[19] Z.T. Beattie, Chad C. Hagen and T.L. Hayes. "Classification of lying position using load cells under the bed". In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. 2011, pp. 474–477. DOI: 10.1109/IEMBS.2011.6090068.

[20] Henk AP Blom and Edwin A Bloem. "Joint Particle Filtering of Multiple Maneuvering Targets From Unassociated Measurements." In: *J. Adv. Inf. Fusion* 1.1 (2006), pp. 15–34.

[21] Harry Blum. "A transformation for extracting new descriptors of shape". In: *Models for the perception of speech and visual form* 19.5 (1967), pp. 362–380.

[22] Johann Borenstein, H. R. Everett and Liqiang Feng. *Navigating Mobile Robots: Systems and Techniques*. Natick, MA, USA: A. K. Peters, Ltd., 1996. ISBN: 1568810660.

[23] R. Bose and A. Helal. "Observing Walking Behavior of Humans Using Distributed Phenomenon Detection and Tracking Mechanisms". In: *Applications and the Internet, 2008. SAINT 2008. International Symposium on*. 2008, pp. 405–408. DOI: 10.1109/SAINT.2008.88.

[24] Andreas Braun, Henning Heggen and Reiner Wichert. "CapFloor - A Flexible Capacitive Indoor Localization System". English. In: *Evaluating AAL Systems Through Competitive Benchmarking. Indoor Localization and Tracking*. Ed. by Stefano Chessa and Stefan Knauth. Vol. 309. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2012, pp. 26–35. ISBN: 978-3-642-33532-7. DOI: 10.1007/978-3-642-33533-4_3.

[25] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern and Steven Shafer. "EasyLiving: Technologies for Intelligent Environments". English. In: *Handheld and Ubiquitous Computing*. Ed. by Peter Thomas and Hans-W. Gellersen. Vol. 1927. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 12–29. ISBN: 978-3-540-41093-5. DOI: 10.1007/3-540-39959-3_2. URL: http://dx.doi.org/10.1007/3-540-39959-3_2.

[26] C. Buragohain, D. Agrawal and S. Suri. "Distributed Navigation Algorithms for Sensor Networks". In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. 2006, pp. 1–10. DOI: 10.1109/INFOCOM.2006.191.

[27] W. Burgard, M. Moors, C. Stachniss and F.E. Schneider. "Coordinated multi-robot exploration". In: *Robotics, IEEE Transactions on* 21.3 (2005), pp. 376–386. ISSN: 1552-3098. DOI: 10.1109/TRO.2004.839232.

[28] Wolfram Burgard, Mark Moors and Frank Schneider. "Collaborative exploration of unknown environments with teams of mobile robots". In: *Advances in plan-based control of robotic agents*. Springer, 2002, pp. 52–70.

[29] Philippe Claude Cattin. "Biometric authentication system using human gait". PhD thesis. Diss., Technische Wissenschaften ETH Zürich, Nr. 14603, 2002.

[30] Subhash Challa, Mark R. Morelande, Darko Musicki and Robin J. Evans. *Fundamentals of object tracking*. 2011.

[31] M Chan, E Campo and D Esteve. "PROSAFE, a multisensory remote monitoring system for the elderly or the handicapped. 1st ICOST". In: *International Conference On Smart homes & health Telematics (ICOST'03), Independent living for persons with disabilities and elderly people, Ed. M. Mokhtari, IOS Press ISBN*. Vol. 1. 380.8. 2003, pp. 89–95.

[32] Seongju Chang, Sungil Ham, Seungbum Kim, Dongjun Suh and Hyunseok Kim. "UbiFloor: Design and Pilot Implementation of an Interactive Floor System". In: *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2010 2nd International Conference on*. Vol. 2. 2010, pp. 290–293. DOI: 10.1109/IHMSC.2010.172.

[33] Duc Phu Chau, F. Bremond and M. Thonnat. "Online evaluation of tracking algorithm performance". In: *Crime Detection and Prevention (ICDP 2009), 3rd International Conference on*. 2009, pp. 1–6. DOI: 10.1049/ic.2009.0266.

[34] H. Choset, I. Konukseven and J. Burdick. "Mobile robot navigation: issues in implementating the generalized Voronoi graph in the plane". In: *Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on*. 1996, pp. 241–248. DOI: 10.1109/MFI.1996.572184.

[35] Marco Contigiani, Emanuele Frontoni, Adriano Mancini and Andrea Gatto. "Indoor people localization and tracking using an energy harvesting smart floor". In: *Mechatronic and Embedded Systems and Applications (MESA), 2014 IEEE/ASME 10th International Conference on*. IEEE. 2014, pp. 1–5.

[36] P Couturier, S Moine, F Favre-Reguillon, Y Caritu, F Giraud-By, R Guillemaud, P Barralon, N Noury, M Berenguer, H Provost *et al.*. "Actidom: Monitoring of activity in frail elderly in their daily life". In: *Proceeding of the iternational meeting on MIcro and NAnoTEChnology, MINATEC2003, Grenoble, France*. 2003.

[37] Hardy Cross. "Analysis of continuous frames by distributing fixed-end moments". In: *American Society of Civil Engineers Transactions* (1932).

[38] S.K. Das, D.J. Cook, A. Battacharya, III Heierman E.O. and Tze-Yun Lin. "The role of prediction algorithms in the MavHome smart home architecture". In: *Wireless Communications, IEEE* 9.6 (2002), pp. 77–84. ISSN: 1536-1284. DOI: 10.1109/MWC.2002.1160085.

[39] Debraj De, Shaojie Tang, Wen-Zhan Song, Diane Cook and Sajal K. Das. "ActiSen: Activity-aware Sensor Network in Smart Environments". In: *Journal of Pervasive and Mobile Computing (PMC)* (2012).

[40] Michael B. Dillencourt, Hanan Samet and Markku Tamminen. "A General Approach to Connected-component Labeling for Arbitrary Image Representations". In: *J. ACM* 39.2 (1992), pp. 253–280. ISSN: 0004-5411. DOI: 10.1145/128749.128750. URL: http://doi.acm.org/10.1145/128749.128750.

[41] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *Robotics & Automation Magazine, IEEE* 13.2 (2006), pp. 99–110.

[42] Gerhard Elger and Barbro Furugren. "SmartBo-an ICT and computer-based demonstration home for disabled people". In: *Proceedings of the 3rd TIDE Congress: Technology for Inclusive Design and Equality Improving the Quality of Life for the European Citizen. Helsinki, Finland June*. 1998.

[43] *Etude sur le marché de l'offre de soins, d'hébergement et de services destinés aux personnes âgées dépendantes*. Tech. rep. Ernst & Young, 2008. URL: http://www.senat.fr/commission/missions/Dependance/etude.pdf.

[44] Ettore Ferranti, Niki Trigoni and Mark Levene. "Brick&Mortar: An Online MultiAgent Exploration Algorithm". In: *IEEE Int. Conf. Robotics and Automation (ICRA)*. 2007.

[45] J Finkelstein, G O'Connor and RH Friedmann. "Development and implementation of the home asthma telemonitoring (HAT) system to facilitate asthma self-care". In: *Studies in health technology and informatics* 84.Pt 1 (2001), 810—814. ISSN: 0926-9630. URL: http://europepmc.org/abstract/MED/11604847.

[46] Martin A. Fischler and Phyllis Barrett. "An iconic transform for sketch completion and shape abstraction". In: *Computer Graphics and Image Processing* 13.4 (1980), pp. 334–360.

[47] Thomas E. Fortmann, Y. Bar-Shalom and M. Scheffe. "Sonar tracking of multiple targets using joint probabilistic data association". In: *Oceanic Engineering, IEEE Journal of* 8.3 (1983), pp. 173–184. ISSN: 0364-9059. DOI: 10.1109/JOE.1983.1145560.

[48] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz and Gaetano Borriello. "Bayesian filtering for location estimation". In: *IEEE pervasive computing* 3 (2003), pp. 24–33.

[49] Linda P. Fried, Catherine M. Tangen, Jeremy Walston, Anne B. Newman, Calvin Hirsch, John Gottdiener, Teresa Seeman, Russell Tracy, Willem J. Kop, Gregory Burke and Mary Ann McBurnie. "Frailty in Older Adults: Evidence for a Phenotype". In: *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences* 56.3 (2001), pp. M146–M157. DOI: 10.1093/gerona/56.3.M146. eprint: http://biomedgerontology.oxfordjournals.org/content/56/3/M146.full.pdf+html.

[50] GAITRite. *GAITRite Portable Walkway System*. URL: http://www.gaitrite.com/surface.htm.

[51] S. Garrido, L. Moreno, M. Abderrahim and F. Martin. "Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. 2006, pp. 2376–2381. DOI: 10.1109/IROS.2006.282649.

[52] Arnaud Glad, Olivier Simonin, Olivier Buffet and François Charpillet. "Theoretical Study of ant-based Algorithms for Multi-Agent Patrolling". Anglais. In: *18th European Conference on Artificial Intelligence including Prestigious Applications of Intelligent Systems (PAIS 2008) - ECAI 2008*. M. Ghallab et al. Patras, Grèce: IOS press, 2008, pp. 626–630. DOI: 10.3233/978-1-58603-891-5-626. URL: http://hal.inria.fr/inria-00326963.

[53] Rupert Glaser, Christl Lauterbach, Domnic Savio, Markus Schnell, Sinan Karadal, Werner Weber, Susanne Kornely and A Stohr. *Smart Carpet: A textile-based large-area sensor network*. 2007.

[54] Fred Glover and Manuel Laguna. *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997. ISBN: 079239965X.

[55] Niall Griffith and Mikael Fernström. "LiteFoot: A floor space for recording dance and controlling media". In: *Proceedings of the 1998 International Computer Music Conference*. 1998, pp. 475–481.

[56] S. Guillen, M.T. Arredondo, V. Traver, J.M. Garcia and C. Fernandez. "Multimedia telehomecare system using standard TV set". In: *Biomedical Engineering, IEEE Transactions on* 49.12 (2002), pp. 1431–1437. ISSN: 0018-9294. DOI: 10.1109/TBME.2002.805457.

[57] S. Helal, B. Winkler, Choonhwa Lee, Y. Kaddoura, L. Ran, C. Giraldo, S. Kuchibhotla and W. Mann. "Enabling location-aware pervasive computing applications for the elderly". In: *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*. 2003, pp. 531–536. DOI: 10.1109/PERCOM.2003.1192785.

[58] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura and E. Jansen. "The Gator Tech Smart House: a programmable pervasive space". In: *Computer* 38.3 (2005), pp. 50 –60. ISSN: 0018-9162. DOI: 10.1109/MC.2005.107.

[59] Ben Heller, Terry Senior and Jon Wheat. "The Smartfloor: A Large Area Force-measuring Floor for Investigating Dynamic Balance and Motivating Exercise". In: *Procedia Engineering* 72.0 (2014). The Engineering of Sport 10, pp. 226 –231. ISSN: 1877-7058. DOI: http://dx.doi.org/10.1016/j.proeng.2014.06.040. URL: http://www.sciencedirect.com/science/article/pii/S1877705814005566.

[60] David J. Hewson, Jacques Duchêne, François Charpillet, Jamal Saboune, Valérie Michel-Pellegrino, Hassan Amoud, Michel Doussot, Jean Paysant, Anne Boyer and Jean-Yves Hogrel. "The PARAChute Project: Remote Monitoring of Posture and Gait for Fall Prevention". In: *EURASIP J. Appl. Signal Process.* 2007.1 (Jan. 2007), pp. 109–109. ISSN: 1110-8657. DOI: 10.1155/2007/27421. URL: http://dx.doi.org/10.1155/2007/27421.

[61] Gang Huang, Hong Jiang, Kim Matthews and Paul Wilford. "Lensless imaging by compressive sensing". In: *arXiv preprint, arXiv:1305.7181* (2013).

[62] Haosheng Huang and Georg Gartner. "A Survey of Mobile Indoor Navigation Systems". English. In: *Cartography in Central and Eastern Europe*. Ed. by Georg Gartner and Felix Ortag. Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg, 2010, pp. 305–319. ISBN: 978-3-642-03293-6. DOI: 10.1007/978-3-642-03294-3_20. URL: http://dx.doi.org/10.1007/978-3-642-03294-3_20.

[63] T. Ikeda, H. Ishiguro and T. Nishimura. "People tracking by fusing different kinds of sensors, floor sensors and acceleration sensors". In: *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*. 2006, pp. 530–535. DOI: 10.1109/MFI.2006.265664.

[64] P. Jallon, B. Dupre and M. Antonakios. "A graph based method for timed up amp; go test qualification using inertial sensors". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. 2011, pp. 689–692. DOI: 10.1109/ICASSP.2011.5946497.

[65] Jin-Woo Jung, Z. Bien, Sang-Wang Lee and T. Sato. "Dynamic-footprint based person identification using mat-type pressure sensor". In: *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*. Vol. 3. 2003, 2937–2940 Vol.3. DOI: 10.1109/IEMBS.2003.1280533.

[66] Jin-Woo Jung, Tomomasa Sato and Zeungnam Bien. "Dynamic footprint-based person recognition method using a hidden markov model and a neural network". In: *International Journal of Intelligent Systems* 19.11 (2004), pp. 1127–1141. ISSN: 1098-111X. DOI: 10.1002/int.20040. URL: http://dx.doi.org/10.1002/int.20040.

[67] Nassim Kaldé, Olivier Simonin and François Charpillet. "Asynchronous Computing of a Discrete Voronoi Diagram on a Cellular Automaton Using 1-Norm: Application to Roadmap Extraction". In: *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on*. 2014, pp. 846–852. DOI: 10.1109/ICTAI.2014.130.

[68] Niklas Karlsson, Mario E Munich, Luis Goncalves, Jim Ostrowski, Enrico Di Bernardo and Paolo Pirjanian. "Core technologies for service robotics". In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 3. IEEE. 2004, pp. 2979–2984.

[69] D. Keren, S. Peleg and R. Brada. "Image sequence enhancement using sub-pixel displacements". In: *Computer Vision and Pattern Recognition, 1988. Proceedings CVPR '88., Computer Society Conference on*. 1988, pp. 742–746. DOI: 10.1109/CVPR.1988.196317.

[70] A.A. Khaliq and A. Saffiotti. "Stigmergy at work: Planning and navigation for a service robot on an RFID floor". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. 2015, pp. 1085–1092. DOI: 10.1109/ICRA.2015.7139311.

[71] Cory D. Kidd, Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth D. Mynatt, Thad Starner and Wendy Newstetter. "The Aware Home: A Living Laboratory for Ubiquitous Computing Research". In: *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*. CoBuild '99. London, UK: Springer-Verlag, 1999, pp. 191–198. ISBN: 3-540-66596-X. URL: http://dl.acm.org/citation.cfm?id=645969.674887.

[72] Kistler force plates. URL: http://www.kistler.com/us/en/index.

[73] Lars Klack, Christian Möllering, Martina Ziefle and Thomas Schmitz-Rode. "Future Care Floor: A Sensitive Floor for Movement Monitoring and Fall Detection in Home Environments". English. In: *Wireless Mobile Communication and Healthcare*. Ed. by JamesC. Lin and KonstantinaS. Nikita. Vol. 55. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2011, pp. 211–218. ISBN: 978-3-642-20866-9. DOI: 10.1007/978-3-642-20865-2_27. URL: http://dx.doi.org/10.1007/978-3-642-20865-2_27.

[74] Sven Koenig, Boleslaw Szymanski and Yaxin Liu. "Efficient and inefficient ant coverage methods". In: *Annals of Mathematics and Artificial Intelligence* 31 (1-4 2001), pp. 41–76. ISSN: 1012-2443.

[75] Richard E. Korf. "Real-time heuristic search". In: *Artif. Intell.* 42 (2-3 1990), pp. 189–211. ISSN: 0004-3702. DOI: 10.1016/0004-3702(90)90054-4. URL: http://dl.acm.org/citation.cfm?id=77754.77756.

[76] I. Korhonen, R. Lappalainen, T. Tuomisto, T. Koobi, V. Pentikainen, M. Tuomisto and V. Turjanmaa. "TERVA: wellness monitoring system". In: *Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE*. Vol. 4. 1998, 1988–1991 vol.4. DOI: 10.1109/IEMBS.1998.746993.

[77] KPMG. *Observatoire des EHPAD*. (French). 2013. URL: https://www.kpmg.com/FR/fr/IssuesAndInsights/ArticlesPublications/Documents/Observatoire-EHPAD-2013-KPMG.pdf.

[78] T. Kuragano, A. Yamaguchi and S. Furukawa. "A Method to Measure Foot Print Similarity for Gait Analysis". In: *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*. Vol. 2. 2005, pp. 816–822. DOI: 10.1109/CIMCA.2005.1631569.

[79] PARK Kwang-Hyun and Z Zenn Bien. "Intelligent sweet home for assisting the elderly and the handicapped". In: *Independent Living for Persons with Disabilities and Elderly People: ICOST'2003: 1st International Conference on Smart Homes and Health Telematics*. Vol. 12. IOS Press. 2003, p. 151.

[80] C. Lauterbach, A. Steinhage and A. Techmer. "Large-area wireless sensor system based on smart textiles". In: *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*. 2012, pp. 1–2. DOI: 10.1109/SSD.2012.6198101.

[81] T Lee, Y Kwon and H Kim. "Smart location tracking system using FSR (Force Sensing Resistor)". In: *International Conference on Artificial Reality and Telexistence*. 2004.

[82] Jaana Leikas, Jussi Mattila, Luc Cluitmans and Timo Urhemaa. "IMS-Intuitive Movement Sensing Method". In: *Proceedings of the Smart Objects Conference, Grenoble, France*. Citeseer. 2003, pp. 200–203.

[83] P. Leusmann, C. Mollering, L. Klack, K. Kasugai, M. Ziefle and B. Rumpe. "Your Floor Knows Where You Are: Sensing and Acquisition of Movement Data". In: *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*. Vol. 2. 2011, pp. 61–66. DOI: 10.1109/MDM.2011.29.

[84] Qun Li, Michael De Rosa and Daniela Rus. "Distributed Algorithms for Guiding Navigation Across a Sensor Network". In: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. MobiCom '03. San Diego, CA, USA: ACM, 2003, pp. 313–325. ISBN: 1-58113-753-2. DOI: 10.1145/938985.939017. URL: http://doi.acm.org/10.1145/938985.939017.

[85] M. Lombardi, A. Pieracci, P. Santinelli, R. Vezzani and R. Cucchiara. "Sensing floors for privacy-compliant surveillance of wide areas". In: *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*. 2013, pp. 105–110. DOI: 10.1109/AVSS.2013.6636624.

[86] *Long-term care services in the United States: 2013 overview*. Tech. rep. U.S. Department of Health and Human Services, Centers for Disease Control and Prevention, National Center for Health Statistics, 2013. URL: http://www.cdc.gov/nchs/data/nsltcp/long_term_care_services_2013.pdf.

[87] Charles F Malacaria. "A thin, flexible, matrix-based pressure sensor". In: *Sensors-the Journal of Applied Sensing Technology* 15.9 (1998), pp. 102–107.

[88] Silvano Martello and Paolo Toth. *Knapsack problems*. Wiley New York, 1990.

[89] Katsunori Matsuoka. "Aware home understanding life activities". In: *Proceedings of 2nd International Conference On Smart Homes and Health Telematic, ICOST*. Vol. 4. 2004, pp. 186–193.

[90] Lisa McElligott, Michelle Dillon, Krispin Leydon, Bruce Richardson, Mikael Fernström and Joseph A. Paradiso. "'ForSe FIElds' - Force Sensors for Interactive Environments". In: *Proceedings of the 4th international conference on Ubiquitous Computing*. UbiComp '02. Göteborg, Sweden: Springer-Verlag, 2002, pp. 168–175. ISBN: 3-540-44267-7. URL: http://dl.acm.org/citation.cfm?id=647988.741477.

[91] Lee Middleton, Alex A. Buss, Alex Bazin and Mark S. Nixon. "A Floor Sensor System for Gait Recognition". In: *Proceedings of the Fourth IEEE Workshop on Automatic Identification Advanced Technologies*. AUTOID '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 171–176. ISBN: 0-7695-2475-3. DOI: 10.1109/AUTOID.2005.2. URL: http://dx.doi.org/10.1109/AUTOID.2005.2.

[92] D.L. Mills. *Network Time Protocol (NTP)*. RFC 958. Obsoleted by RFCs 1059, 1119, 1305. Internet Engineering Task Force, 1985. URL: http://www.ietf.org/rfc/rfc958.txt.

[93] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit *et al.*. "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In: *AAAI/IAAI*. 2002, pp. 593–598.

[94] T. Mori, Y. Suemasu, H. Noguchi and Tomomasa Sato. "Multiple people tracking by integrating distributed floor pressure sensors and RFID system". In: *Systems, Man and Cybernetics, 2004 IEEE International Conference on*. Vol. 6. 2004, 5271–5278 vol.6. DOI: 10.1109/ICSMC.2004.1401031.

[95] Taketoshi Mori and Tomomasa Sato. "Robotic room: Its concept and realization". In: *Robotics and Autonomous Systems* 28.2-3 (1999), pp. 141–148. DOI: http://dx.doi.org/10.1016/S0921-8890(99)00012-3.

[96] H. Morishita, R. Fukui and T. Sato. "High resolution pressure sensor distributed floor for future human-robot symbiosis environments". In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. Vol. 2. 2002, 1246 –1251 vol.2. DOI: 10.1109/IRDS.2002.1043914.

[97] Michael C Mozer. "The neural network house: An environment hat adapts to its inhabitants". In: *Proc. AAAI Spring Symp. Intelligent Environments*. 1998, pp. 110–114.

[98] T. Murakita, T. Ikeda and H. Ishiguro. "Human tracking using floor sensors based on the Markov chain Monte Carlo method". In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 4. 2004, 917 –920 Vol.4. DOI: 10.1109/ICPR.2004.1333922.

[99] K. Nagatani, Y. Iwai and Y. Tanaka. "Sensor based navigation for car-like mobile robots using generalized Voronoi graph". In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. Vol. 2. 2001, 1017–1022 vol.2. DOI: 10.1109/IROS.2001.976302.

[100] K. Nakajima, Y. Mizukami, K. Tanaka and T. Tamura. "Footprint-based personal recognition". In: *Biomedical Engineering, IEEE Transactions on* 47.11 (2000), pp. 1534–1537. ISSN: 0018-9294. DOI: 10.1109/10.880106.

[101] Y. Nishida, T. Hori, T. Suehiro and S. Hirai. "Sensorized environment for self-communication based on observation of daily human behavior". In: *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*. Vol. 2. 2000, 1364–1372 vol.2. DOI: 10.1109/IROS.2000.893211.

[102] H. Noguchi, T. Mori and Tomomasa Sato. "Construction of network system and the first step of summarization for human daily action data in the sensing room". In: *Knowledge Media Networking, 2002. Proceedings. IEEE Workshop on*. 2002, pp. 17–22. DOI: 10.1109/KMN.2002.1115157.

[103] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Vol. 501. John Wiley & Sons, 2009.

[104] Robert J. Orr and Gregory D. Abowd. "The smart floor: a mechanism for natural user identification and tracking". In: *CHI '00 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '00. The Hague, The Netherlands: ACM, 2000, pp. 275–276. ISBN: 1-58113-248-4. DOI: 10.1145/633292.633453. URL: http://doi.acm.org/10.1145/633292.633453.

[105] Joseph Paradiso, Craig Abler, Kai-yuh Hsiao and Matthew Reynolds. "The magic carpet: physical sensing for immersive environments". In: *CHI '97 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '97. Atlanta, Georgia: ACM, 1997, pp. 277–278. ISBN: 0-89791-926-2. DOI: 10.1145/1120212.1120391. URL: http://doi.acm.org/10.1145/1120212.1120391.

[106] Shmuel Peleg, Danny Keren and Limor Schweitzer. "Improving image resolution using subpixel motion". In: *Pattern Recognition Letters* 5.3 (1987), pp. 223 –226. ISSN: 0167-8655. DOI: http://dx.doi.org/10.1016/0167-8655(87)90067-5. URL: http://www.sciencedirect.com/science/article/pii/0167865587900675.

[107] Nicolas Pepin, Olivier Simonin and François Charpillet. "Intelligent Tiles: Putting Situated Multi-Agents Models in Real World". Anglais. In: *International Conference on Agents and Artificial Intelligence - ICAART'09*. Ed. by ACM AAAI. Porto, Portugal, 2009. URL: http://hal.inria.fr/inria-00339231.

[108] Russell F. Pinkston. "A touch sensitive dance floor/MIDI controller". In: *The Journal of the Acoustical Society of America* 96.5 (1994), pp. 3302–3302. DOI: http://dx.doi.org/10.1121/1.410820. URL: http://scitation.aip.org/content/asa/journal/jasa/96/5/10.1121/1.410820.

[109] S. Pirttikangas, J. Suutala, J. Riekki and J. Röning. "Learning Vector Quantization in Footstep Identification". In: *Proceedings of 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA 2003)*. Ed. by M.H. Hamza. IASTED. Benalmadena, Spain: ACTA Press, 2003, pp. 413–417.

[110] Susanna Pirttikangas, Jaakko Suutala, Jukka Riekki and Juha Röning. "Footstep identification from pressure signals using Hidden Markov Models". In: *Proc. Finnish Signal Processing Symposium (FINSIG'03)*. 2003, pp. 124–128.

[111] Prima team, Inria. *Casper project*. URL: https://raweb.inria.fr/rapportsactivite/RA2010/prima/uid91.html.

[112] A. Pronobis and P. Jensfelt. "Large-scale semantic mapping and reasoning with heterogeneous modalities". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. 2012, pp. 3515–3522. DOI: 10.1109/ICRA.2012.6224637.

[113] Pulsar team, Inria. *GerhomeProject*. URL: http://www-sop.inria.fr/members/Francois.Bremond/topicsText/gerhomeProject.html.

[114] Anies Hannawati Purnamadjaja and R. Andrew Russell. "Pheromone communication in a robot swarm: necrophoric bee behaviour and its replication". In: *Robotica* 23.6 (Nov. 2005), pp. 731–742. ISSN: 0263-5747. DOI: 10.1017/S0263574704001225. URL: http://dx.doi.org/10.1017/S0263574704001225.

[115] Gang Qian, Jiqing Zhang and Assegid Kidané. "People Identification Using Gait Via Floor Pressure Sensing and Analysis". English. In: *Smart Sensing and Context*. Ed. by Daniel Roggen, Clemens Lombriser, Gerhard Tröster, Gerd Kortuem and Paul Havinga. Vol. 5279. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 83–98. ISBN: 978-3-540-88792-8. DOI: 10.1007/978-3-540-88793-5_7.

[116] Gang Qian, Jiqing Zhang and A. Kidané. "People Identification Using Floor Pressure Sensing and Analysis". In: *Sensors Journal, IEEE* 10.9 (2010), pp. 1447–1460. ISSN: 1530-437X. DOI: 10.1109/JSEN.2010.2045158.

[117] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler and Andrew Y Ng. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. 2009, p. 5.

[118] R. Rajalingham, Y. Visell and J.R. Cooperstock. "Probabilistic Tracking of Pedestrian Movements via In-Floor Force Sensing". In: *Computer and Robot Vision (CRV), 2010 Canadian Conference on*. 2010, pp. 143–150. DOI: 10.1109/CRV.2010.26.

[119] Sankar Rangarajan, Assegid Kidané, Gang Qian, Stjepan Rajko and David Birchfield. "The Design of a Pressure Sensing Floor for Movement-Based Human Computer Interaction". English. In: *Smart Sensing and Context*. Ed. by Gerd Kortuem, Joe Finney, Rodger Lea and Vasughi Sundramoorthy. Vol. 4793. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 46–61. ISBN: 978-3-540-75695-8. DOI: 10.1007/978-3-540-75696-5_3.

[120] Sankar Rangarajan, Assegid Kidané, Gang Qian and Stjepan Rajko. *Design Optimization of Pressure Sensing Floor for Multimodal Human-Computer Interaction*. INTECH Open Access Publisher, 2008. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.164.246.

[121] Mohsen Razeghi and Mark Edward Batt. "Foot type classification: a critical review of current methods". In: *Gait & Posture* 15.3 (2002), pp. 282 –291. ISSN: 0966-6362. DOI: http://dx.doi.org/10.1016/S0966-6362(01)00151-5. URL: http://www.sciencedirect.com/science/article/pii/S0966636201001515.

[122] R.E. Reilly, M.R. Amirinia and R.W. Soames. "A two-dimensional imaging walkway for gait analysis". In: *Computer-Based Medical Systems, 1991. Proceedings of the Fourth Annual IEEE Symposium*. 1991, pp. 145–152. DOI: 10.1109/CBMS.1991.128957.

[123] Bruce Richardson, Krispin Leydon and Mikael Fernström. "Z-Tiles: building blocks for modular, pressure-sensing floorspaces". In: *In CHI '04: CHI '04*. Press, 2004, pp. 1529–1532. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.68.2026.

[124] H Rimminen, J Lindström and R Sepponen. "Positioning accuracy and multi-target separation with a human tracking system using near field imaging". In: *International Journal On Smart Sensing and Intelligent Systems* 2.1 (2009), pp. 156–175.

[125] H. Rimminen, J. Lindström, M. Linnavuo and R. Sepponen. "Detection of Falls Among the Elderly by a Floor Sensor Using the Electric Near Field". In: *Information Technology in Biomedicine, IEEE Transactions on* 14.6 (2010), pp. 1475 –1476. ISSN: 1089-7771. DOI: 10.1109/TITB.2010.2051956. URL: http://dx.doi.org/10.1109/TITB.2010.2051956.

[126] Maxime Rio, Francis Colas, Mihai Andries and François Charpillet. *Probabilistic sensor data processing for robot localisation on load-sensing floors*. (submitted to) IEEE International Conference on Robotics and Automation (ICRA 2016).

[127] P. Robert, E. Castelli, P.-C. Chung, T. Chiroux, C.F. Crispim-Junior, P. Mallea and F. Bremond. "SWEET-HOME {ICT} technologies for the assessment of elderly subjects". In: *{IRBM}* 34.2 (2013). Special issue : {ANR} {TECSAN} : Technologies for Health and Autonomy, pp. 186 –190. ISSN: 1959-0318. DOI: http://dx.doi.org/10.1016/j.irbm.2013.01.015. URL: http://www.sciencedirect.com/science/article/pii/S1959031813000298.

[128] MJ Rodriguez, MT Arredondo, F Del Pozo, EJ Gomez, A Martinez and A Dopico. "A home telecare management system". In: *Journal of telemedicine and telecare* 1.2 (1995), pp. 86–94.

[129] M. Rohrbach, S. Amin, M. Andriluka and B. Schiele. "A database for fine grained activity detection of cooking activities". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. 2012, pp. 1194–1201. DOI: 10.1109/CVPR.2012.6247801.

[130] Antti Ropponen, Matti Linnavuo and Raimo Sepponen. "LF indoor location and identification system". In: *International Journal on Smart Sensing and Intelligent System* 2.1 (2009), pp. 94–117.

[131] Antti Ropponen, Henry Rimminen and Raimo Sepponen. "Robust System for Indoor Localisation and Identification for the Health Care Environment". English. In: *Wireless Personal Communications* 59.1 (2011), pp. 57–71. ISSN: 0929-6212. DOI: 10.1007/s11277-010-0189-z. URL: http://dx.doi.org/10.1007/s11277-010-0189-z.

[132] Azriel Rosenfeld and John L. Pfaltz. "Sequential operations in digital picture processing". In: *Journal of the ACM (JACM)* 13.4 (1966), pp. 471–494.

[133] R. Andrew Russell. "Laying and sensing odor markings as a strategy for assisting mobile robot navigation tasks". In: *Robotics& Automation Magazine, IEEE* 2 (3 1995), pp. 3–9. ISSN: 1070-9932. DOI: http://dx.doi.org/10.1109/100.414920.

[134] R. Andrew Russell. "Heat trails as short-lived navigational markers for mobile robots". In: *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*. Albuquerque, 1997, pp. 3534–3539.

[135] R.A. Russell, D. Thiel and A. Mackay-Sim. "Sensing odour trails for mobile robot navigation". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. San Diego, 1994, pp. 2672–2677.

[136] T. Sato, T. Harada and T. Mori. "Environment-type robot system "RoboticRoom" featured by behavior media, behavior contents, and behavior adaptation". In: *Mechatronics, IEEE/ASME Transactions on* 9.3 (2004), pp. 529 –534. ISSN: 1083-4435. DOI: 10.1109/TMECH.2004.834650.

[137] Domnic Savio and Thomas Ludwig. "Smart Carpet: A Footstep Tracking Interface". In: *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 02*. AINAW '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 754–760. ISBN: 0-7695-2847-3. DOI: 10.1109/AINAW.2007.338. URL: http://dx.doi.org/10.1109/AINAW.2007.338.

[138] Albrecht Schmidt, Martin Strohbach, Kristof van Laerhoven, Adrian Friday and Hans-Werner Gellersen. "Context Acquisition Based on Load Sensing". In: *Proceedings of the 4th international conference on Ubiquitous Computing*. UbiComp '02. Göteborg, Sweden: Springer-Verlag, 2002, pp. 333–350. ISBN: 3-540-44267-7. URL: http://dl.acm.org/citation.cfm?id=647988.741476.

[139] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, ArnoldP. Boedihardjo, Crystal Chen, Susan Frankenstein and Manfred Lerner. "GrammarViz 2.0: A Tool for Grammar-Based Pattern Discovery in Time Series". English. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Toon Calders, Floriana Esposito, Eyke Hüllermeier and Rosa Meo. Vol. 8726. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 468–472. ISBN: 978-3-662-44844-1. DOI: 10.1007/978-3-662-44845-8_37. URL: http://dx.doi.org/10.1007/978-3-662-44845-8_37.

[140] Titus Sharpe and Barbara Webb. "Simulated and situated models of chemical trail following in ants". In: *Proceedings of the fifth international conference on simulation of adaptive behavior on From animals to animats 5*. Univ. of Zurich, Zurich, Switzerland: MIT Press, 1998, pp. 195–204. ISBN: 0-262-66144-6. URL: http://dl.acm.org/citation.cfm?id=299955.299980.

[141] Yu-Lin Shen and Chow-Shing Shin. "Distributed Sensing Floor for an Intelligent Environment". In: *Sensors Journal, IEEE* 9.12 (2009), pp. 1673 –1678. ISSN: 1530-437X. DOI: 10.1109/JSEN.2009.2030650.

[142] *Silver Economie*. URL: http://www.social-sante.gouv.fr/espaces,770/personnes-agees-autonomie,776/dossiers,758/silver-economie,2432/.

[143] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun and Håkan Younes. "Coordination for multi-robot exploration and mapping". In: *AAAI/IAAI*. 2000, pp. 852–858.

[144] Olivier Simonin, Thomas Huraux and François Charpillet. "Interactive Surface for Bio-inspired Robotics, Re-examining Foraging Models". Anglais. In: *23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. Boca Raton, USA: IEEE, 2011.

[145] D. Slater and F. Mitchell. *High precision cartesian robot for a planar scanner*. US Patent App. 09/873,907. 2002. URL: https://www.google.com/patents/US20020180393.

[146] M. Sousa, A. Techmer, A. Steinhage, C. Lauterbach and P. Lukowicz. "Human tracking and identification using a sensitive floor and wearable accelerometers". In: *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. 2013, pp. 166–171. DOI: 10.1109/PerCom.2013.6526728.

[147] Prashant Srinivasan, David Birchfield, Gang Qian and Assegid Kidané. "A pressure sensing floor for interactive media applications". In: *In Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE), 2005. (Ref.* 2005, p. 26.

[148] Prashant Srinivasan, David Birchfield, Gang Qian and Assegid Kidané. "Design of a Pressure Sensitive Floor for Multimodal Sensing". In: *Proceedings of the Ninth International Conference on Information Visualisation*. IV '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 41–46. ISBN: 0-7695-2397-8. DOI: 10.1109/IV.2005.40. URL: http://dx.doi.org/10.1109/IV.2005.40.

[149] A. Sud, E. Andersen, S. Curtis, M.C. Lin and D. Manocha. "Real-Time Path Planning in Dynamic Virtual Environments Using Multiagent Navigation Graphs". In: *Visualization and Computer Graphics, IEEE Transactions on* 14.3 (2008), pp. 526–538. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.27.

[150] J. Suutala and J. Roning. "Combining classifiers with different footstep feature sets and multiple samples for person identification". In: *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*. Vol. 5. 2005, v/357–v/360 Vol. 5. DOI: 10.1109/ICASSP.2005.1416314.

[151] J. Suutala, K. Fujinami and J. Roning. "Persons tracking with Gaussian process joint particle filtering". In: *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*. 2010, pp. 160–165. DOI: 10.1109/MLSP.2010.5589263.

[152] Jaakko Suutala and Juha Röning. "Towards the adaptive identification of walkers: Automated feature selection of footsteps using distinction-sensitive LVQ". In: *Int. Workshop on Processing Sensory Information for Proactive Systems (PSIPS 2004)*. 2004, pp. 14–15.

[153] Jaakko Suutala and Juha Röning. "Methods for person identification on a pressure-sensitive floor: Experiments with multiple classifiers and reject option". In: *Information Fusion* 9.1 (2008). Special Issue on Applications of Ensemble Methods, pp. 21 –40. ISSN: 1566-2535. DOI: http://dx.doi.org/10.1016/j.inffus.2006.11.003. URL: http://www.sciencedirect.com/science/article/pii/S156625350600114X.

[154] Jaakko Suutala, Kaori Fujinami and Juha Röning. "Gaussian Process Person Identifier Based on Simple Floor Sensors". English. In: *Smart Sensing and Context*. Ed. by Daniel Roggen, Clemens Lombriser, Gerhard Tröster, Gerd Kortuem and Paul Havinga. Vol. 5279. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 55–68. ISBN: 978-3-540-88792-8. DOI: 10.1007/978-3-540-88793-5_5.

[155] Jonas Svennebring and Sven Koenig. "Building Terrain-Covering Ant Robots: A Feasibility Study". In: *Auton. Robots* 16.3 (May 2004), pp. 313–332. ISSN: 0929-5593. DOI: 10.1023/B:AURO.0000025793.46961.f6. URL: http://dx.doi.org/10.1023/B:AURO.0000025793.46961.f6.

[156] Tarkett. *FloorInMotion*. URL: http://www.floorinmotion.com/.

[157] TechStorm foot scan. URL: http://techstorm.koreasme.com/pro/pro09.html.

[158] AM. Tekalp, M.K. Ozkan and M.I Sezan. "High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration". In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. Vol. 3. 1992, 169–172 vol.3. DOI: 10.1109/ICASSP.1992.226249.

[159] Tekscan pressure mapping, force measurement and tactile sensors. URL: https://www.tekscan.com/product-group/medical/mats-and-walkways?tab=products-solutions.

[160] The Active Gaming Company. *Lightspace floor*. URL: http://www.the-agc.com/products/physical-therapy/lightspace-floor/.

[161] Jean-Pierre Thomesse. "TIISSAD, Technologies de l'information intégrées aux services des soins à domicile". In: *Conférence Journées AIM Télémédecines et technologies de Santé*. Colloque avec actes et comité de lecture. nationale. La Pitié Salpétrière/France, June 2001, 13 p. URL: https://hal.inria.fr/inria-00100642.

[162] Jean-Pierre Thomesse, François Charpillet, Luis Véga, Pierre-Yves Durand and Jacques Chanliau. "Diatélic() : un système intelligent de télémédecine appliqué à la dialyse: aspects médicaux, informatiques, économiques". In: *2ème Conférence Francophone en Gestion et Ingénierie des Systèmes Hospitaliers - GISEH'2004*. Colloque sur invitation. Internationale. Mons/Belgique, 2004. URL: https://hal.inria.fr/inria-00100067.

[163] S. Thrun and A. Bücken. "Integrating Grid-Based and Topological Maps for Mobile Robot Navigation". In: *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*. Portland, Oregon, 1996.

[164] Sebastian Thrun. "The Role of Exploration in Learning Control". In: *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Ed. by D.A. White and D.A. Sofge. Florence, Kentucky 41022: Van Nostrand Reinhold, 1992.

[165] Sebastian Thrun, Dieter Fox, Wolfram Burgard and Frank Dellaert. "Robust Monte Carlo localization for mobile robots". In: *Artificial intelligence* 128.1 (2001), pp. 99–141.

[166] B.C. Tom and AK. Katsaggelos. "Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images". In: *Image Processing, 1995. Proceedings., International Conference on*. Vol. 2. 1995, 539–542 vol.2. DOI: 10.1109/ICIP.1995.537535.

[167] Yu-Chee Tseng, Meng-Shiuan Pan and Yuen-Yung Tsai. "Wireless sensor networks for emergency navigation". In: *Computer* 39.7 (2006), pp. 55–62. ISSN: 0018-9162. DOI: 10.1109/MC.2006.248.

[168] Panagiotis G Tzionas, Adonios Thanailakis and Philippos G Tsalides. "Collision-Free Path Planning for a Diamond-Shaped Robot Using Two-Dimensional Cellular Automata". In: *IEEE Transactions on Robotics* 13.2 (1997), pp. 237–250.

[169] M. Valtonen, J. Maentausta and J. Vanhala. "TileTrack: Capacitive human tracking using floor tiles". In: *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. 2009, pp. 1 –10. DOI: 10.1109/PERCOM.2009.4912749.

[170] R. Vera-Rodriguez, J.S.D. Mason, J. Fierrez and J. Ortega-Garcia. "Analysis of Time Domain Information for Footstep Recognition". English. In: *Advances in Visual Computing*. Ed. by George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Ronald Chung, Riad Hammoud, Muhammad Hussain, Tan Kar-Han, Roger Crawfis, Daniel Thalmann, David Kao and Lisa Avila. Vol. 6453. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 489–498. ISBN: 978-3-642-17288-5. DOI: 10.1007/978-3-642-17289-2_47.

[171] R. Vera-Rodriguez, J.S.D. Mason, J. Fierrez and J. Ortega-Garcia. "Comparative Analysis and Fusion of Spatiotemporal Information for Footstep Recognition". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.4 (2013), pp. 823–834. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.164.

[172] Rubén Vera-Rodríguez, Richard P Lewis, John SD Mason and Nicholas WD Evans. "Footstep recognition for a smart home environment". In: *International Journal of Smart Home* 2.2 (2008), pp. 95–110.

[173] Rubén Vera-Rodriguez, JohnS.D. Mason and NicholasW.D. Evans. "Assessment of a Footstep Biometric Verification System". English. In: *Handbook of Remote Biometrics*. Ed. by Massimo Tistarelli, StanZ. Li and Rama Chellappa. Advances in Pattern Recognition. Springer London, 2009, pp. 313–327. ISBN: 978-1-84882-384-6. DOI: 10.1007/978-1-84882-385-3_13.

[174] Atul Verma, Hemjit Sawant and Jindong Tan. "Selection and navigation of mobile sensor nodes using a sensor network". In: *Pervasive and Mobile Computing* 2.1 (2006), pp. 65–84.

[175] Cor Vermeulen and Ad van Berlo. "Exchange on Housing". In: *Gerontechnology: a sustainable investment in the future* 48 (1998), p. 337.

[176] Y. Visell, A. Law and J.R. Cooperstock. "Touch Is Everywhere: Floor Surfaces as Ambient Haptic Interfaces". In: *Haptics, IEEE Transactions on* 2.3 (2009), pp. 148–159. ISSN: 1939-1412. DOI: 10.1109/TOH.2009.31.

[177] Y. Visell, S. Smith, A. Law, R. Rajalingham and J.R. Cooperstock. "Contact sensing and interaction techniques for a distributed, multimodal floor display". In: *3D User Interfaces (3DUI), 2010 IEEE Symposium on*. 2010, pp. 75–78. DOI: 10.1109/3DUI.2010.5444718.

[178] Yon Visell, JeremyR. Cooperstock, BrunoL. Giordano, Karmen Franinovic, Alvin Law, Stephen McAdams, Kunal Jathal and Federico Fontana. "A Vibrotactile Device for Display of Virtual Ground Materials in Walking". English. In: *Haptics: Perception, Devices and Scenarios*. Ed. by Manuel Ferre. Vol. 5024. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 420–426. ISBN: 978-3-540-69056-6. DOI: 10.1007/978-3-540-69057-3_55. URL: http://dx.doi.org/10.1007/978-3-540-69057-3_55.

[179] Yon Visell, Alvin Law, Jessica Ip, Severin Smith and Jeremy R. Cooperstock. "Interaction capture in immersive virtual environments via an intelligent floor surface". In: *Virtual Reality Conference (VR), 2010 IEEE*. 2010, pp. 313 –314. DOI: 10.1109/VR.2010.5444748.

[180] Israel A. Wagner, Michael Lindenbaum and Alfred M. Bruckstein. "Distributed covering by ant-robots using evaporating traces". In: *IEEE Transactions on Robotics and Automation* 15 (1999), pp. 918–933.

[181] Jiliang Wang, Zhenjiang Li, Mo Li, Yunhao Liu and Zheng Yang. "Sensor Network Navigation without Locations". In: *Parallel and Distributed Systems, IEEE Transactions on* 24.7 (2013), pp. 1436–1446. ISSN: 1045-9219. DOI: 10.1109/TPDS.2012.207.

[182] Liau Wen-Hau, Wu Chao-Lin and Fu Li-Chen. "Inhabitants Tracking System in a Cluttered Home Environment Via Floor Load Sensors". In: *Automation Science and Engineering, IEEE Transactions on* 5.1 (2008), pp. 10 –20. ISSN: 1545-5955. DOI: 10.1109/TASE.2007.911671.

[183] P. Willett, Yanhua Ruan and R. Streit. "PMHT: problems and some solutions". In: *Aerospace and Electronic Systems, IEEE Transactions on* 38.3 (2002), pp. 738–754. ISSN: 0018-9251. DOI: 10.1109/TAES.2002.1039396.

[184] G. Williams, Kevin Doughty and D.A. Bradley. "A systems approach to achieving CarerNet-an integrated and intelligent telecare system". In: *Information Technology in Biomedicine, IEEE Transactions on* 2.1 (1998), pp. 1–9. ISSN: 1089-7771. DOI: 10.1109/4233.678527.

[185] David A Winter. *Biomechanics and motor control of human gait: normal, elderly and pathological*. Waterloo, Ont: University of Waterloo Press, 1991.

[186] *World Population Prospects, the 2015 Revision*. 2015. URL: http://esa.un.org/unpd/wpp/DVD/.

[187] XSensor technology corporation. *XSensor*. 2002. URL: http://xsensor.com/.

[188] B. Yamauchi. "A frontier-based approach for autonomous exploration". In: *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. 1997, pp. 146–151. DOI: 10.1109/CIRA.1997.613851.

[189] Zhenwang Yao and Kamal Gupta. "Distributed roadmaps for robot navigation in sensor networks". In: *Robotics, IEEE Transactions on* 27.5 (2011), pp. 997–1004.

[190] KangKang Yin and Dinesh K. Pai. "FootSee: An Interactive Animation System". In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '03. San Diego, California: Eurographics Association, 2003, pp. 329–338. ISBN: 1-58113-659-5. URL: http://dl.acm.org/citation.cfm?id=846276.846323.

[191] Chen-Rong Yu, Chao-Lin Wu, Ching-Hu Lu and Li-Chen Fu. "Human Localization via Multi-Cameras and Floor Sensors in Smart Home". In: *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*. Vol. 5. 2006, pp. 3822–3827. DOI: 10.1109/ICSMC.2006.384726.

[192] Jae-Seok Yun, Seung-Hum Lee, Woon-Tack Woo and Je-Ha Ryu. "The user identification system using walking pattern over the ubifloor". In: *Proceedings of International Conference on Control, Automation, and Systems*. Vol. 1046. 2003, p. 1050.

[193] Jaeseok Yun. "User identification using gait patterns on UbiFloorII". In: *Sensors* 11.3 (2011), pp. 2611–2639.

[194] Jaeseok Yun, Woontack Woo and Jeha Ryu. "User Identification Using User's Walking Pattern over the ubiFloorII". English. In: *Computational Intelligence and Security*. Ed. by Yue Hao, Jiming Liu, Yuping Wang, Yiu-ming Cheung, Hujun Yin, Licheng Jiao, Jianfeng Ma and Yong-Chang Jiao. Vol. 3801. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 949–956. ISBN: 978-3-540-30818-8. DOI: 10.1007/11596448_141. URL: http://dx.doi.org/10.1007/11596448141.

[195] Jaeseok Yun, Gregory D. Abowd, Jeha Ryu and Woontack Woo. "User identification with user's stepping pattern over the ubiFloorII". In: *International Journal of Pattern Recognition and Artificial Intelligence* 22.03 (2008), pp. 497–514. DOI: 10.1142/S0218001408006338. eprint: http://www.worldscientific.com/doi/pdf/10.1142/S0218001408006338. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218001408006338.

Glossary

ANR Agence Nationale de la Recherche (French National Research Agency). 9

BMI Brick&Mortar Improved. 101, 110, 112–116

BMILRV Brick&Mortar Improved with long range vision. 101, 116, 118, 120, 121

BMIS Brick&Mortar Improved Simplified. 113–115

BMS Brick&Mortar Simplified. 112, 113, 115

BOCDA Brillouin optical correlation domain analysis. 17

CoDyCo Whole-Body Compliant Dynamical Contacts for Cognitive Humanoids. 9

COP center of pressure. 22, 26, 35, 44, 83, 84

CPER Contrat de plan État-Région (planning agreement between the national and regional governments). 9

DSLVQ Distinction-sensitive Learning Vector Quantization. 21, 26

EKF Extended Kalman filter. 38

EMFi electro-mechanical film. 18, 21, 26

FLD Fisher linear discriminant. 26

FSR force-sensing resistor. 18, 20–26

GRF ground reaction force. 23

HMM Hidden Markov Model. 20–22, 26, 73, 74

ICRA International Conference on Robotics and Automation. 38, 62, 128

IP address Internet Protocol address. 33

IROS IEEE/RSJ International Conference on Intelligent Robots and Systems. 102

JSON JavaScript Object Notation. 98

KF Kalman filter. 35, 38

LAR Living Assistant Robot. 9

LED Light-emitting diode. 32, 42, 43

LiDAR light detection and ranging. 91

LRTA* Learning Real-Time A*. 112

LVQ Learning Vector Quantization. 21, 26

MAIA Machines Intelligentes Autonomes. 7

MCMC Markov Chain Monte Carlo. 19

MDF medium-density fibreboard. 25

MLP Multi-layer perceptron. 26

N/A not available. 18

Network Time Protocol a networking protocol used to synchronize timekeeping among a set of distributed time servers and clients . 49

OHS Office d'Hygiène Sociale. 8

PAL Personally Assisted Living. 8, 9

PDA Probability Data Association. 23

PIR pyroelectric infrared sensors. 16, 25

RFID radio-frequency identification. 16, 24, 27, 94

ROS Robot Operating System. 32, 48, 49, 98, 128

SLAM Simultaneous Localization and Mapping. 91

SVM Support Vector Machine. 23, 26

TRIO Temps réel et interopérabilité (Real time and interoperability). 7

Unix time The number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), 1 January 1970, not counting leap seconds.. 33

List of Figures

List of Tables

List of Algorithms

Résumé

Cette thèse explore les capacités d'une intelligence ambiante équipée d'un réseau de capteurs de pression au sol. Elle traite le problème de la perception d'un environnement au travers un réseau de capteurs de basse résolution. Les difficultés incluent l'interpretation des poids dispersés pour des objets avec multiples supports, l'ambiguïté de poids entre des objets, la variation du poids des personnes pendant les activités dynamiques, etc. Nous introduisons des nouvelles techniques, partiellement inspirées du domaine de la vision par l'ordinateur, pour la détection, le suivi et la reconnaissance des entités qui se trouvent sur le sol. Nous introduisons également des nouveaux modes d'interaction entre les environnements équipés de tels capteurs aux sol, et les robots qui évoluent dans ces environnements. Ceci permet l'interpretation non-intrusive des événements qui ont lieu dans des environnements dotés d'une intelligence ambiante, avec des applications dans l'assistance automatisée à domicile, l'aide aux personnes âgées, le diagnostique continu de la santé, la sécurité, et la navigation robotique.

Mots-clés: intelligence ambiante, réseau de capteurs au sol, détection de pression à haute résolution, suivi, localisation, reconnaissance des objets, navigation robotique, exploration multi-robot distribuée, stigmergie.

Abstract

This thesis explores the capabilities of an ambient intelligence equipped with a load-sensing floor. It deals with the problem of perceiving the environment through a network of low-resolution sensors. Challenges include the interpretation of spread loads for objects wit multiple points of support, weight ambiguities between objects, variation of persons' weight during dynamic activities, etc. We introduce new techniques, partly inspired from the field of computer vision, for detecting, tracking and recognising the entities located on the floor. We also introduce new modes of interaction between environments equipped with such floor sensors and robots evolving inside them. This enables non-intrusive interpretation of events happening inside environments with embedded ambient intelligence, with applications in assisted living, senile care, continuous health diagnosis, home security, and robotic navigation.

Keywords: ambient intelligence, sensing floors, high-resolution pressure sensing, tracking, localisation, recognition, robotic navigation, distributed multi-robot exploration, stigmergy.